ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIONES

Integración de Inteligencia Artificial en un Sistema Embebido para Medidas Optoeléctricas

Adecuación y mejora del sistema de medición optoeléctrico

Autor: Adrián Vicente Gómara

Director: César Elosúa Aguado

Pamplona, 18 Marzo 2016

RESUMEN

Este proyecto consiste en la implementación de una red neuronal en el sistema embebido PIC 16F877A de la familia Microchip para realizar medidas de absorción óptica y clasificar distintas muestras. Gracias a algoritmo de la red se puede identificar el tipo de muestra que se está analizando a partir de la medición de la absorbancia para tres longitudes de onda (rojo, verde y azul). El sistema en conjunto se ha integrado en una placa de circuito impreso (PCB), simplificando la circuitería y componentes propuestos en un diseño previo. El código se ha programado en C y se ha optimizado para utilizar los recursos de memoria disponibles en el PIC, haciendo un uso eficiente de los mismos. El tiempo de ejecución es lo suficientemente bajo como para poder realizar la medida y clasificación de la muestra en pocos segundos, por lo que se puede considerar un sistema a tiempo real. Los resultados obtenidos animan a la mejora del prototipo desarrollado para su aplicación en la práctica.

Palabras Clave: *Red Neuronal Artificial, Medidas Absorbancia, PIC, Programación C*

ABSTRACT

The project describes the implementation of an Artificial Neural Network in the embedded system PIC 16F877A from Microchip to perform optical absorbance measurements and classify liquid samples. Due to the network algorithm, the analyzed sample can be classified. Light Emiting Diodes (LEDs) at three different wavelengths (Red, Green and Blue) are used to record three different absorbance measurements, using them to identify the sample by the Neural Network. The whole system has been integrated on a Printed Circuit Board (PCB), optimizing the components used on a previous design. The code has been programed and optimized in C in order to use in an efficient way the memory resources of the PIC. Execution time is low enough to allow measurements to be made on real time. The obtained results encourage the improvement of the prototype in future works to use it in real applications

Key Words: *Artificial Neural Network, Absorbance Measurements, PIC, C Programming*

Índice

Capítulo 1: Introducción y Objetivos

1.1 Antecedentes

Este proyecto consiste en la implementación de una red neuronal perceptron multicapa en un sistema optoeléctrico de medición a varias longitudes de onda cuya finalidad es obtener la absorbancia y su posterior caracterización determinada por la red. En él se explica el desarrollo del sistema emisor de luz que se basa en diodos LED, el receptor y el procesamiento de la señal de medida obtenida, así como la interfaz de visualización de datos y ajuste de potencias.

Además se realizarán una serie de simplificaciones y mejoras del diseño previo en cuanto al hardware y software para una mejor integración de la aplicación.

1.2 Introducción a las Redes Neuronales

Las redes neuronales (ANN) son técnicas no paramétricas muy utilizadas en diversos ámbitos de la ciencia e ingeniería porque permiten resolver problemas complejos, que muchas veces no son fáciles de resolver utilizando técnicas tradicionales como la regresión lineal o polinómica. Las redes neuronales permiten obtener un modelo no explícito que relaciona un conjunto de variables salida con un conjunto de variables entrada. Así, estos modelos permiten predecir cuál es el valor de salida, dados unos valores de entrada del modelo. Para estimar el modelo es necesario disponer de un conjunto de observaciones de las variables.

Estas observaciones son usadas como patrones de entrenamiento para que la red aprenda y sea capaz de predecir una salida del modelo, ante nuevas observaciones. Por tanto, las capacidades de la red van a depender en gran medida de esta fase de entrenamiento. En la fase de entrenamiento es necesario controlar muchos parámetros y distintos algoritmos de optimización, por lo que el usuario de una red neuronal debe tener conocimiento suficiente de cuáles son estos parámetros y cómo funcionan. Por otro lado, una vez entrenada la red, es muy importante también evaluar la robustez del modelo creado, comprobando que es adecuado para nuevos datos. Es importante, realizar un buen análisis de los resultados obtenidos.

Existen muchos tipos diferentes de redes neuronales, pero en este proyecto sólo se trabajará con la red backpropagation (o perceptron multicapa) que es muy utilizada en la práctica.

1.3 Medidas de absorbancia y su aplicación

Desde la antigüedad el ser humano ha mostrado interés en el estudio y comprensión de la luz. Nuestro ojo es capaz de ver solo un pequeño rango del espectro de esta, pero está compuesta por muchas más frecuencias de las que podemos ver.

El estudio de la luz comienza con el experimento del prisma de Isaac Newton, este consistía en hacer pasar un haz de luz blanca, no necesariamente procedente del sol, por un prisma y ver que este se descomponía en diferentes haces de diferente color, precisamente los del arcoíris. Más tarde se comprobó que cada color tenía una frecuencia electromagnética diferente.

En los siglos XVIII y XIX se pudo mejorar este prisma con lentes y rendijas y se consiguió una herramienta mejor para la medida de la luz. Joseph Von Fraunhofer captó la luz solar con esta herramienta y vio que había pequeñas franjas oscuras en el espectro. A partir de aquí se analizaron las luces de diferentes materiales al ser calentados y se pudo ver que cada uno emitía unas bandas de frecuencia diferentes. Con esto se pudo analizar cada material y se creó una “huella digital” de los diferentes elementos.

También se descubrió que calentando suficientemente un material hasta ponerlo incandescente, este producía una luz blanca con el espectro sin ninguna banda oscura. Si se hacía pasar esta luz por un fino material y se captaba la luz que pasaba por este material se observó que en el espectro había unas bandas negras precisamente en las frecuencias en las que emitía luz si se calentaba este mismo material. Este es el origen del estudio de la transmitancia y la absorbancia en los distintos materiales.

Para medir la absorbancia de la materia se utiliza un dispositivo denominado espectrofotómetro que hace incidir diferentes longitudes de onda a una muestra donde se es capaz de ver la cantidad de energía electromagnética que absorbe para cada frecuencia. Con esta parametrización se es posible identificar el tipo de muestra gracias a las redes neuronales.

1.4 Sistema Embebido

Un sistema embebido (anglicismo de embedded) o empotrado (integrado, incrustado) es un sistema de computación diseñado para realizar una o pocas funciones dedicadas que se diseñan para cubrir necesidades específicas. El componente principal del sistema es el microprocesador donde se programan las funciones a realizar.

El microprocesador que se va a utilizar es un PIC de la familia 16F, en concreto el PIC 16F877A, que posee las siguientes características:

- Memoria de programa: FLASH, 8 K de instrucciones de 14 bits c/u.
- Memoria de datos: 368 bytes RAM, 256 bytes EEPROM.

- Pila (Stack): 8 niveles (14 bits).
- Fuentes de interrupción : 13
- Instrucciones: 35
- Encapsulado: DIP de 40 pines.
- Frecuencia oscilador: 20 MHz (máxima)
- Temporizadores/Contadores: 1 de 8 bits (Timer 0); 1 de 16 bits (Timer 1); 1 de 8 bits (Timer 2) con pre y post escalador. Un perro guardián (WDT)
- Líneas de E/S : 6 del puerto A, 8 del puerto B, 8 del puerto C, 8 del puerto D y 3 del puerto E, además de 8 entradas análogas.
- Dos módulos de Captura, Comparación y PWM: - Captura: 16 bits. Resolución máx. = 12.5 nseg. - Comparación: 16 bits. Resolución máx. = 200 nseg. - PWM: Resolución máx. = 10 bits.
- Convertidor Análogo/Digital de 10 bits multicanal (8 canales de entrada).
- Puerto serial síncrono (SSP) con bus SPI (modo maestro) y bus I^2C (maestro/esclavo).
- USART (Universal Synchronous Asynchronous Receiver Transmitter) con dirección de detección de 9 bits.
- Corriente máxima absorbida/suministrada (sink/source) por línea (pin): 25 mA
- Oscilador: Soporta 4 configuraciones diferentes: XT, RC, HS, LP.
- Tecnología de Fabricación: CMOS
- Voltaje de alimentación: 3.0 a 5.5 V DC
- Puede operar en modo microprocesador.

1.5 Objetivos

En este proyecto se desarrollara una aplicación para medir la absorbancia de muestras a diferentes longitudes de onda junto con la integración de inteligencia artificial en un PIC de tal manera que se obtenga una mejor comprensión de los resultados y cierta autonomía de decisión en base al aprendizaje del dispositivo. Debido a las especificaciones de memoria del PIC la depuración del código va a ser un elemento esencial ya que marcara las características de la red.

Con esta aplicación se pretende introducirse en los sistemas de inteligencia artificial para proporcionar una mejor interface máquina-humano, cuyo fin es mejorar las relaciones entre humanos y así una mejor integración de la especie con sí misma y con su entorno.

- Aplicar el código de la ANN que utiliza Matlab al PIC.
- Mejorar el diseño previo del espectrofotómetro y eliminar aquellas partes que no se utilizan.
- Adecuar el código del prototipo previo para cumplir con los requisitos de aplicación y de memoria exigidos por el PIC para la integración de la ANN.

Capítulo 2: Fundamentos Teóricos de las redes neuronales

2.1 Fundamentos biológicos de las redes neuronales

Las redes neuronales artificiales se basan en el funcionamiento del sistema neuronal del cuerpo humano. En el cuerpo humano encontramos 3 elementos fundamentales: los órganos receptores que recogen información del exterior; el sistema nervioso que transmite la información, la analiza y en parte almacena, y envía la información elaborada y, los órganos efectores que reciben la información de parte del sistema nervioso y la convierte en una cierta acción.

La unidad fundamental del sistema nervioso es la neurona. Las neuronas se unen unas con otras formando redes. Se componen de un cuerpo o núcleo, del axón, que es una ramificación de salida de la neurona, y de un gran número de ramificaciones de entrada llamadas dendritas. Su funcionamiento es el siguiente. Las señales de entrada llegan a la neurona a través de la sinapsis, que es la zona de contacto entre neuronas (u otro tipo de células, como las receptoras). La sinapsis recoge información electro-química procedente de las células adyacentes que están conectadas a la neurona en cuestión. Esta información llega al núcleo de la neurona, a través de las dendritas, que la procesa hasta generar una respuesta, la cual es posteriormente propagada por el axón.

La sinapsis está compuesta de un espacio líquido donde existe una cierta concentración de iones. Este espacio tiene unas determinadas características eléctricas que permiten inhibir o potenciar la señal eléctrica a conveniencia.

Por ello, se puede ver que el sistema neuronal es un conjunto de neuronas conectadas entre sí, que reciben, elaboran y transmiten información a otras neuronas, y que dicha información se ve potenciada o inhibida en la siguiente neurona a conveniencia, gracias a las propiedades del espacio intersináptico.

De hecho, esta propiedad de poder alterar el peso de cada información en la red neuronal nos otorga en cierta medida la capacidad de aprender.

2.2 Redes neuronales artificiales (ANN):

Las redes neuronales artificiales tratan de emular las características y propiedades de las redes neuronales biológicas. En general, consisten en una serie de unidades denominadas neuronas, conectadas entre sí.

Cada neurona recibe un valor de entrada, el cual transforma según una función específica denominada función de activación. Dicha señal transformada pasa a ser la salida de la neurona.

Las neuronas se conectan entre sí según una determinada arquitectura. Cada conexión tiene un determinado peso que pondera cada entrada a la neurona. De esta manera la entrada de cada neurona es la suma de las salidas de las neuronas conectadas a ella, multiplicadas por el peso de la respectiva conexión. La figura siguiente ilustra dicho concepto:

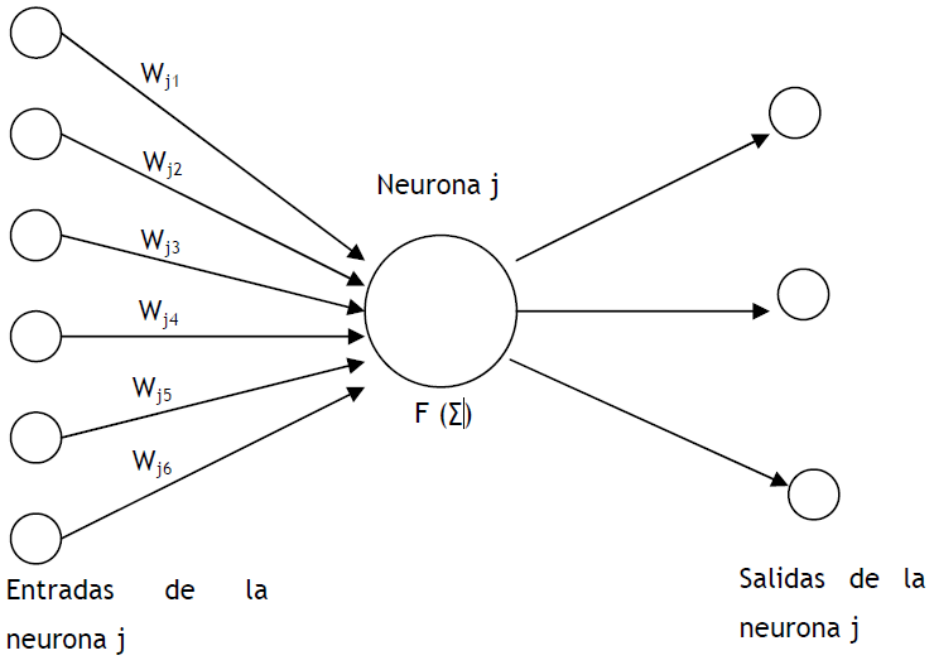


Figura 1: Modelo de Red Neuronal

En este modelo, la neurona j recibe una serie de entradas x1, x2, …, xn. Cada señal se multiplica por el peso asociado a su conexión, w1, w2,…,wn. Luego, se suman estas entradas ponderadas y se les aplica la función de activación F(.) para generar la señal de salida de la neurona j. Los valores de los pesos son ajustados durante la fase de aprendizaje.

Como se ha comentado anteriormente, estas neuronas están conectadas entre sí de acuerdo a una determinada arquitectura. Es decir, las neuronas se agrupan en distintas capas: una capa de entrada, otra de salida, y en el caso de existir, una o varias capas ocultas. La salida de cada neurona se propaga por igual por estas conexiones hasta las neuronas de destino. Cada conexión tiene un peso asociado que pondera el valor numérico de la señal que viaja por ésta. Así pues, una red de neuronas artificial puede verse como un grafo cuyos nodos tienen funcionamiento similar, los cuales propagan la información a través de las distintas conexiones.

Veamos el funcionamiento de una red. Para ello nos referimos a la figura 2 Las entradas a la red son introducidas en las neuronas de la capa de entrada, que normalmente genera una salida tal cual o las escala para que las señales se encuentren en un determinado rango. Estas entradas son propagadas a las

neuronas de la siguiente capa. De acuerdo al esquema de la figura 1 cada neurona j de la segunda capa generará una salida de valor:

$$S_{2j} = F_{2j}W_{1j}$$

Ecuación 1

Donde X1 es el vector de entradas de la capa 1 y W1j el vector de pesos correspondientes a las conexiones que van de todas las neuronas de la primera capa a la neurona j de la segunda capa. La función F2j es la función de activación de la neurona j de la segunda capa. Así con todas las neuronas de la segunda capa. Estas salidas son propagadas a las neuronas de la capa de salida. Estas neuronas generan las salidas de la red. Cada neurona i de la capa de salida generará una salida de valor:

$$S_{si} = F_{si}(W_{2i}S_2)$$

Ecuación 2

Donde W2i es el vector de pesos correspondientes a las conexiones que van de las neuronas de la segunda capa a la neurona i de la capa de salida, y S2 el vector de salidas de las neuronas de la capa dos, que a su vez son entradas de las neuronas de la capa de salida.

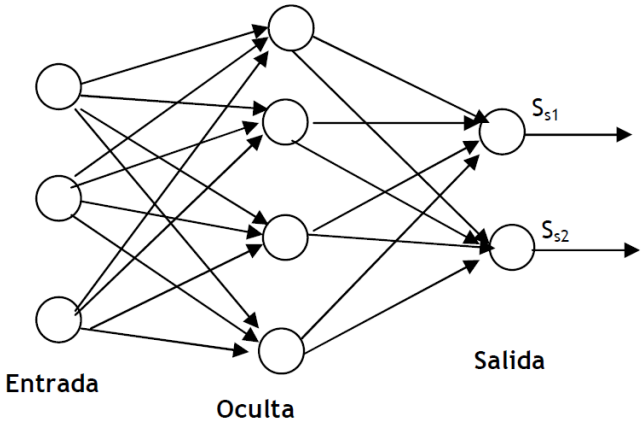


Figura 2

Por último hablar del aspecto más importante y delicado de redes neuronales, el aprendizaje. Las RNA son sistemas de aprendizaje basadas en datos que son utilizados como patrones. Por ello la capacidad de una red de resolver un problema está muy ligada a los patrones utilizados durante su fase de aprendizaje.

El aprendizaje de una red neuronal consiste en hallar los valores precisos de los pesos de sus conexiones para que pueda resolver un determinado problema. El proceso general consiste en ir introduciendo una serie de datos patrón y ajustar los pesos siguiendo un determinado criterio. Los criterios que se van a utilizar en este proyecto se fundamentan en el error cometido por la red, lo

que nos obliga a conocer la salida que se debería obtener para cada uno de ellos. Es lo que se conoce como entrenamiento supervisado. De esta manera, primero se introducen los patrones, se reajustan los pesos, posteriormente se comprueba si se ha cumplido un determinado criterio de convergencia, de no ser así se repite todo el proceso.

2.3 Redes Perceptron Multicapa

2.3.1 Introducción

El perceptron multicapa con conexiones hacia adelante es una generalización del perceptron simple. Surge como respuesta a los problemas que tenía dicha red, como por ejemplo, no poder resolver problemas que no fueran linealmente separables. De hecho, algunos autores han demostrado que el perceptron multicapa es un aproximador universal de cualquier función continua en el espacio $\mathbb{R}$.

2.3.2 Arquitectura del perceptron multicapa

La arquitectura de este tipo de red se caracteriza porque tiene todas sus neuronas agrupadas en distintos niveles llamados capas. El primer nivel corresponde a la capa de entrada, que se encarga únicamente de propagar por el resto de la red las entradas recibidas. El último nivel es el de la capa de salida. Se encarga de proporcionar los valores de salida de la red. En las capas intermedias denominadas capas ocultas, se realiza un procesamiento no lineal de los patrones recibidos.

Las conexiones del perceptron multicapa son hacia adelante. Generalmente todas las neuronas de un nivel se conectan con todas las neuronas de la capa inmediatamente posterior. A veces, dependiendo de la red, se encuentran conexiones de neuronas que no están en niveles consecutivos, o alguna de las conexiones entre dos neuronas de niveles consecutivos no existe, es decir, el peso asociado a dicha conexión es constante e igual a cero. Además, todas las neuronas de la red tienen un valor umbral asociado. Se suele tratar como una entrada cuyo valor es constante e igual a uno, y lo único que varía es el peso asociado a dicha conexión (que es el umbral realmente). Por otro lado, la función de activación que se utilizara es la función sigmoidal:

$$f(x) = \frac{1}{1+e^{-x}}$$

Ecuación 3

donde su rango es [0,1].

Capítulo 3: Materiales y Dispositivos

3.1 Material Químico

Para la realización de las muestras se parte de dos disoluciones madre:

- 500 ml de agua pura y 66 mg de Congo Red.
- 500 ml de agua pura y 20 mg de Methiline blue.

Con cada una de las disoluciones madre se realizan muestras de 150 ml con diferentes concentraciones:

- 25%, se utilizan 37,5 ml de la madre y 112,5 ml de agua pura.
- 50%, se utilizan 75 ml de la madre y 75 ml de agua pura.
- 75%, se utilizan 112,5 ml de agua pura y 37,5 ml de la madre.
- 100%, se emplean 150 ml de la madre.

Con cada una de estas concentraciones se rellenan cubetas de 5 ml que se introducen en el portacubetas.

3.1.1 Materiales utilizados para conseguir las muestras

- Cubetas de 5ml.
- Jeringuillas de 5ml
- 2 Vasos de 500ml y 8 vasos de 200 ml.
- 1Bascula
- Pato de plástico para pesar.
- Agitador magnético.
- Guantes para la manipulación y bata blanca.
- Marcador y etiquetas de muestras.
- Papel absorbente.

Figura 3: material y equipo químico utilizado

3.2 Materiales Electrónicos

Los materiales a utilizar en este proyecto son principalmente componentes electrónicos para montaje en PCB. Se han escogido bien por criterio del propio diseño del dispositivo o porque para el montaje de alguno de ellos en la hoja de características el fabricante proporciona un circuito recomendado para su correcto funcionamiento. La mayoría de componentes más específicos como el fotodiodo o el microprocesador se han comprado vía online en las páginas de RS Components, Mouser y Farnell. Los elementos más sencillos como resistencias o potenciómetros se han obtenido de la clase de electrónica básica del departamento de ingeniería eléctrica-electrónica de la UPNA.

A continuación se realizará una descripción de los elementos clave del dispositivo.

3.2.1 Microprocesador PIC

El microprocesador que se ha escogido es un PIC de la familia 16F, en concreto el PIC 16F877A. El dispositivo cuenta con ocho módulos de conversión analógico-digital de 10 bit, que se utilizarán para recoger la señal procedente del fotodiodo. Cuenta con entradas y salidas suficientes para satisfacer las necesidades requeridas, en concreto cuenta con cuatro puertos y cada puerto cuenta con 8 pines de entrada o salida configurables vía software, a excepción del puerto A que solamente cuenta con 6 pines. El oscilador interno con el que cuenta el micro es de 20MHz, pero pueden acoplarse otros osciladores externos para poder trabajar a diferentes frecuencias.

El PIC16F877 se basa en la arquitectura Harvard, en la cual el programa y los datos se pueden trabajar con buses y memorias separadas, lo que posibilita que las instrucciones y los datos posean longitudes diferentes.

La memoria de programa es una memoria flash de 8K de longitud con datos de 14 bits en cada posición. Como es del tipo FLASH se puede programar y borrar eléctricamente, lo que facilita el desarrollo de los programas y la experimentación. En ella se graba o almacena el programa o códigos que el microcontrolador debe ejecutar. En la figura 4 se muestra el mapa de la memoria de programa. Está se divide en cuatro bancos o páginas de 2K cada uno. El primero va de la posición de memoria 0000h a la 07FFh, el segundo va de la 0800h a la 0FFFh, el tercero de la 1000h a la 17FFh y el cuarto de la 1800h a la 1FFFh.

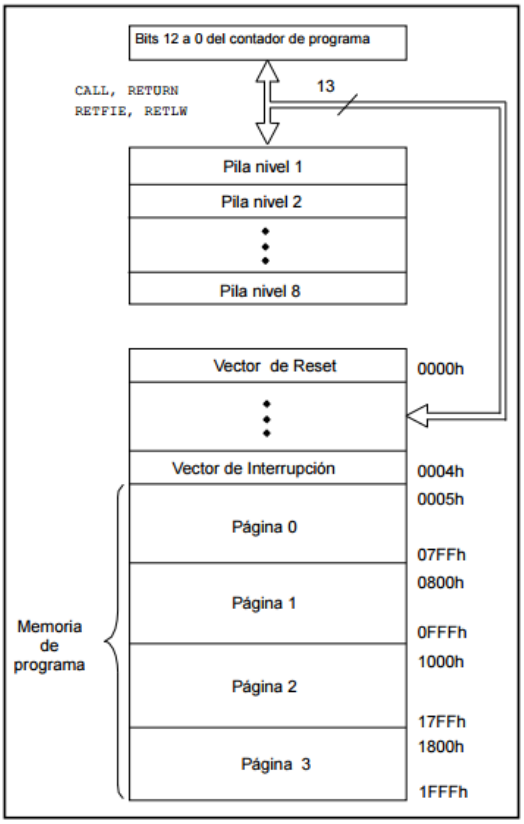


Figura 4: Bancos de la memoria de programa

La memoria de datos posee cuatro bancos de memoria RAM, cada banco posee 128 bytes. De estos 128 los primeros 32 (hasta el 1Fh) son registros que cumplen un propósito especial en el control del microcontrolador y en su configuración. Los 96 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se está ejecutando, en la figura 5 se pueden ver los distintos bancos.

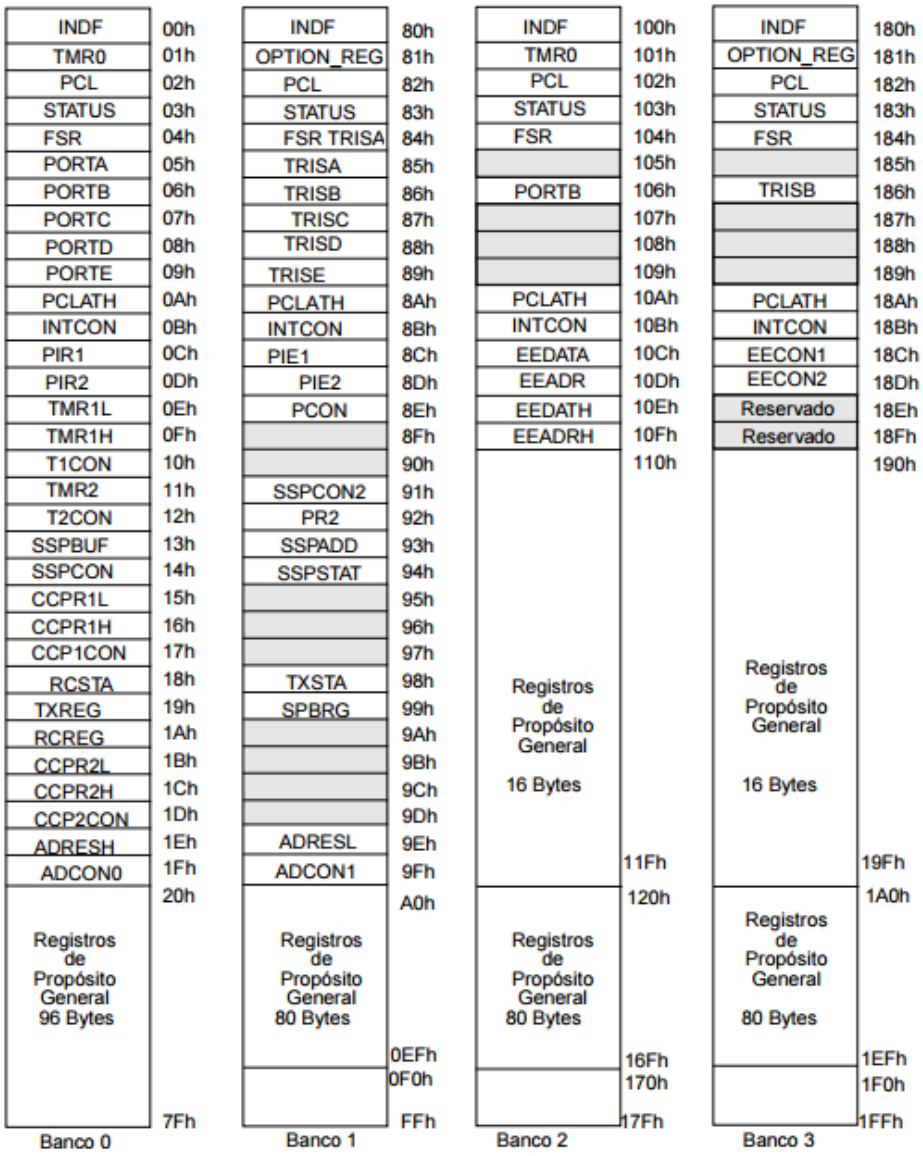


Figura 5: Bancos de memoria RAM

06h o PORTB: Puerto de entrada/salida de 8 bits. Al igual que en todos los PIC, este puede leerse o escribirse como si se tratara de un registro cualquiera; algunos de sus pines tienen funciones alternas en la generación de interrupciones. El registro de control para la configuración de la función de sus pines se localiza en la página 1, en la dirección 86h y se llama TRISB. Puede ser configurado también para cumplir otras funciones.

07h o PORTC: Puerto de entrada/salida de 8 bits. Al igual que en todos los PIC, este puede leerse o escribirse como si se tratara de un registro cualquiera; algunos de sus pines tienen funciones alternas. El registro de control para la configuración de la función de sus pines se localiza en la página 1, en la dirección 87h y se llama TRISC. Puede ser configurado también para cumplir otras funciones.

Registros del módulo de A/D:

- ADRESH : Parte alta del resultado de la conversión
- ADRESL: Parte baja del resultado de la conversión
- ADCON0: Registro de Control 0 ;control del funcionamiento del conversor
- ADCON1, Registro de Control 1; configuración de los pines del puerto

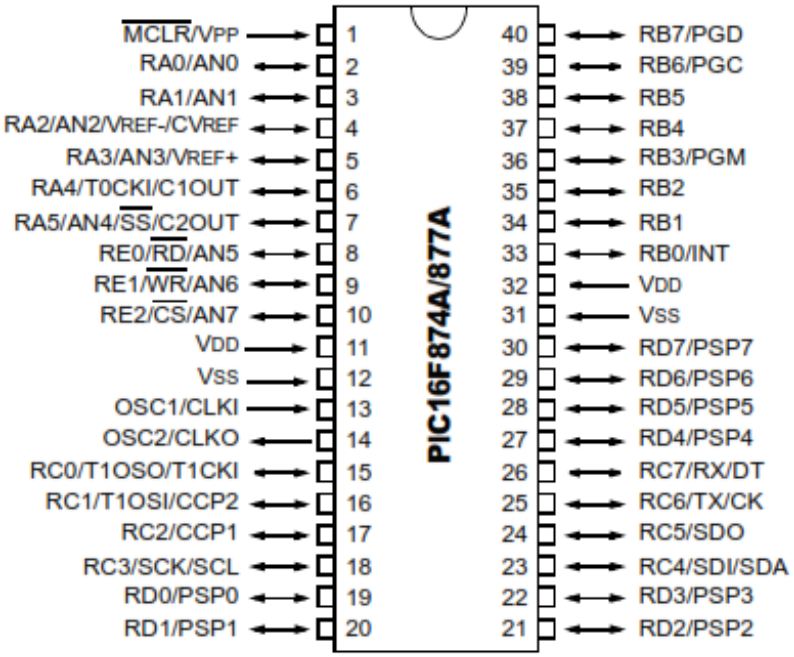


Figura 6: PIC 16F877A.

REGISTRO ADCON0 (DIRECCIÓN LFH)

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	-	ADON
bit7							bit 0

Figura 7

bit 7-6: **ADCS1:ADCS0**: En estos dos bits se hace la selección de la frecuencia de reloj para el Convertidor A/D.

00	Fosc/2
01	Fosc /8
10	Fosc/32
11	FRC (Procede del oscilador RC interno)

Tabla 1

bit 5-3: **CH2:CH0**: Aquí se selecciona el canal analógico por donde entrará la señal a digitalizar. En este microcontrolador tenemos 8 canales de entrada al Conversor A/D

000	Canal 0, (RA0/AN0)
001	Canal 1, (RA1/AN1)
010	Canal 2, (RA2/AN2)
011	Canal 3, (RA3/AN3)
100	Canal 4, (RA4/AN4)
101	Canal 5, (RA5/AN5)
110	Canal 6, (RA6/AN6)
111	Canal 7, (RA7/AN7)

Tabla 2

bit 2: **GO/#DONE**. bit de estado de la conversión A/D

Si ADON=1

1= La conversión A/D está en marcha (mientras está a 1 se está realizando la conversión)

0 = La conversión ha finalizado. (El bit se pone a cero automáticamente por hardware cuando la conversión A/D finaliza) el resultado de la conversión aparece en ADRESH: ADRESL

bit 1: **No implementado**: Se lee como “0”

bit 0: **ADON**: bit de puesta en marcha

1 = El convertidor A/D está operativo

0 = El convertidor A/D está apagado y no consume corriente.

El registro ADCON1

El registro ADCON1 es uno de los registros del convertidor A/D del PIC16F877, se trata de un registro de configuración de los pines del puerto, este registro se compone de 8 bits, los cuales describamos su función a continuación:

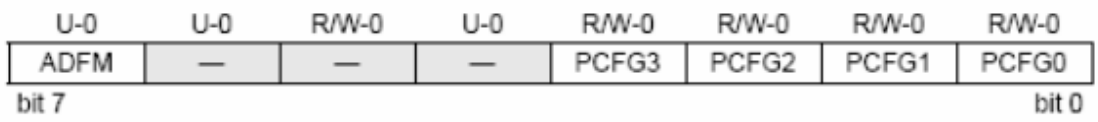


Figura 8

Bit 7: ADFM: Selecciona el formato del resultado de la conversión A/D

1 = >Pone en el registro ARDESH los seis bits de mayor peso a “0”

0 = >Pone los 6 bits de menor peso del registro ADRESL a “0”

Bits 6-4: No implementados: Se leen como cero

Bit 3-0: PCFG3:PCFG0: bits de configuración de los canales de entrada del convertidor A/D. Se utilizan para configurar las patillas como E/S digital o como entrada analógica de acuerdo con la tabla 9.

PCFG3: PCFG0	AN7(1) RE2	AN6(1) RE1	AN5(1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN / REFS
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Entrada Análoga

D = Entrada/Salida Digital

Figura 9

LOS REGISTROS ADRESH Y ADRESL

El par de registros **ADRESH: ADRESL** se carga con el resultado de 10 bits de la conversión A/D. Este par de registros se extienden hasta 16 bits. El módulo A/D tiene la posibilidad de justificar el resultado de 10 bits dentro de los 16 bits de la pareja de registros. La selección del formato de justificación a la izquierda o derecha se realiza con el bit ADFM (**ADCON1**). Los bits restantes (a los 10 de la conversión) se llenan con ceros.

Estos dos registros cuando el convertidor A/D está en OFF y no se utiliza, pueden utilizarse como dos registros de 8 bits de propósito general. Cuando se completa la conversión A/D, el resultado se guarda en los registros y se pone a cero el bit **GO/DONE**

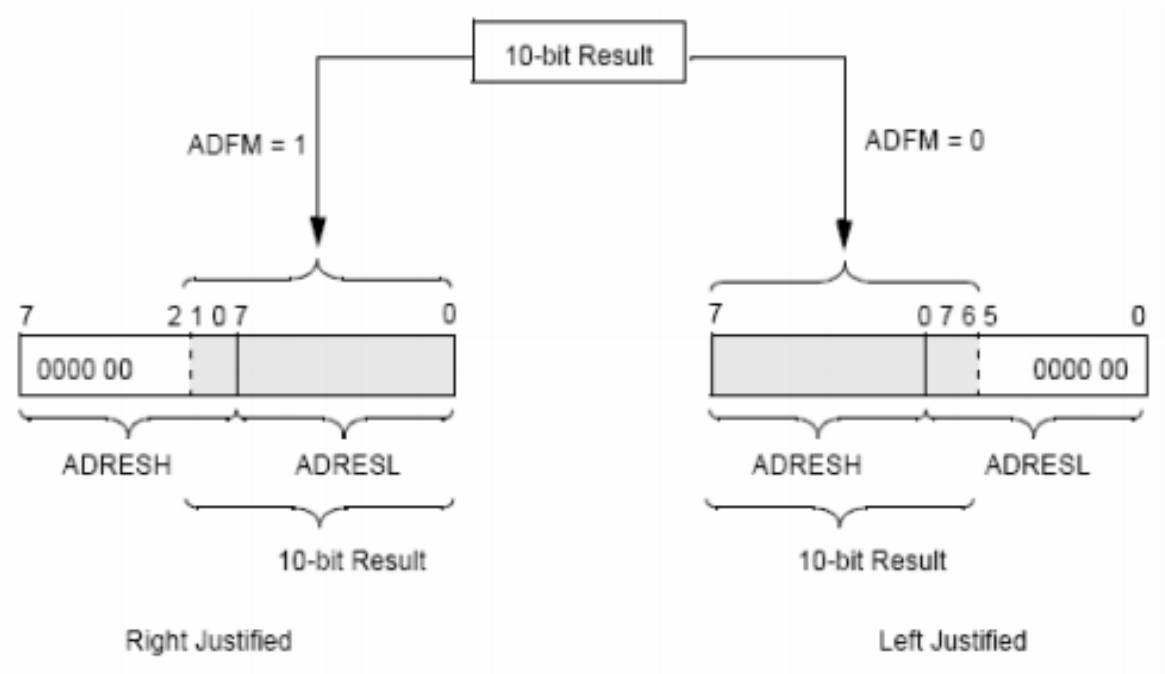


Figura 10

Por lo tanto, los 16 bits que forman el registro **ARDESH-ARDESL** con **ADFM**=1 tiene los 6 bits de mayor peso a cero y con **ADFM**=0 los 6 bit de menor peso están a cero, en los 10 bits restantes se almacena el resultado de la conversión.

3.2.2 Fotodiodo

El PIN-6DI es un fotodiodo de silicio difuso plano diseñado para aplicaciones de alta velocidad y sensibilidad. Tiene un rango de captación de espectro de 350 a 1100nm. Tiene una responsividad típica de 0.65 A/w en su punto máximo que se encuentra a 970nm y cuenta con un área activa de 16,5mm^2. La combinación de estas dos características hace que la corriente generada por este fotodiodo sea aceptable para la aplicación final.

Figura 11: Fotodiodo PIN-6DI

Dos de los aspectos que se deben tener en cuenta según la aplicación que se le quiera dar al dispositivo son la capacidad parásita y la corriente generada en oscuridad total. Estos parámetros responden de diferente manera al voltaje de polarización inversa. En el caso de la capacidad, a más voltaje de

polarización, esta disminuye y con la corriente oscura ocurre lo contrario, a más voltaje más corriente. Esta última también aumenta con la temperatura.

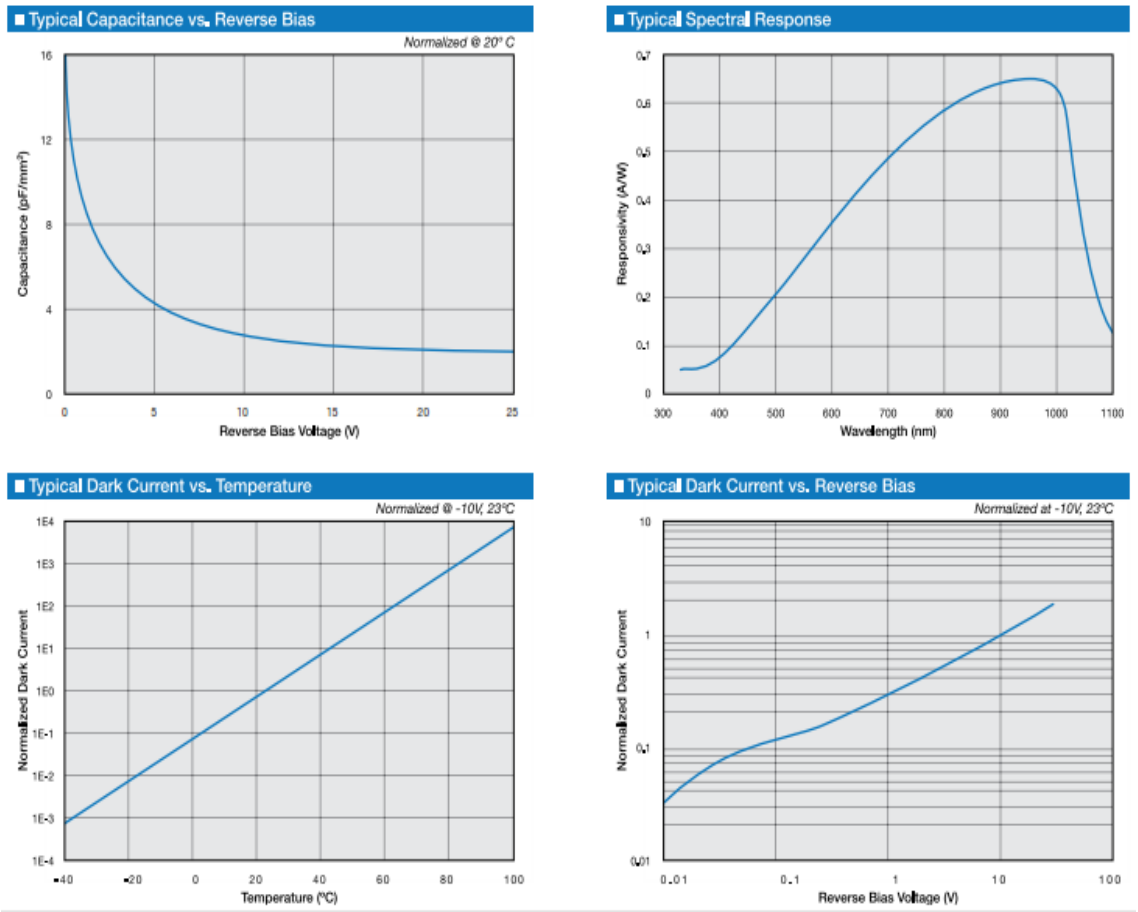


Figura 12: Curvas de trabajo del fotodiodo

3.2.3 Pantalla LCD

El dispositivo para mostrar los datos es una pantalla LCD de 16 caracteres por línea y dos líneas cuyo fabricante es MIKROE-55. Utiliza el controlador Hitachi HD44780 que es el mismo que utiliza el PIC, lo que hace que sean compatibles con las librerías que se utilizan.

Figura 13: Pantalla LCD 932 MIKROE 55

3.2.4 Convertidor Digital-Analógico

Convertidor digital analógico, DAC0808LCN/NOPB, 8 bits-Bit MDIP, 16-Pines Paralelo. Este tiene una resolución de 8 bit y tiene entrada paralela por la cual llega el número binario procedente del PIC que lo convierte a una corriente proporcional. En este dispositivo es posible ajustar el rango de salida mediante las patillas Vref+ y Vref-. Con esto se podrá ajustar la variación de voltaje a la salida por cada bit.

Figura 14: Convertidor digital-analógico

3.2.5 Diodo led tricolor kingbright

Se han escogido intentando abarcar diferentes partes del rango visible del espectro electromagnético. De este modo las longitudes de onda elegidas son 630 nm, 465 nm y 525 nm. Todo en un encapsulado de 5mm de diámetro. En el apartado anexo se puede ver su datasheet.

Figura 15: Led tricolor Kingbright

3.2.6 Fuente de alimentación

MCEXT5V15WC1 Es una fuente de la empresa Multicomp. Tiene un rango de entrada de 90-230V de corriente alterna. La salida es de 5v, lo que hace que no haya que colocar ningún conversor DC/DC a la salida de la fuente para alimentar al circuito, ya que todos los componentes van alimentados a este mismo voltaje. Tiene una potencia de 12W y no precisa de carga mínima.

Figura 16: Fuente de alimentación Multicomp.

3.2.7 PCB

La Placa base está fabricada con una fresa de control numérico, que a partir de una placa de cobre de PCB, se realiza el archivo diseñado con desingspark el cuál se convierte a formato Gerber (estándar fresado).

3.2.8 Componentes electrónicos menos relevantes

- Conector RJ12
- 2 Interruptores táctil tipo Botón, Negro, contactos SPST
- Condensador electrolítico de aluminio Panasonic ECA1AHG472, 4700µF, ±20%, 10 V dc, Serie NHG
- Inductor de montaje en superficie bobinado Murata, 10A Idc, Serie BNX002
- Controlador PWM, MAX735EPA+, -4,75 V 275 mA PDIP 8 pines 160 kHz Ajustable, fijo
- 2 Amplificadores operacionales MCP6292-E/P, 3 V, 5 V 10MHz CMOS, Rail to Rail PDIP, 8 pines
- Inductor radial Bourns, 10 µH ±10%, Ferrita, SRF máxima:16MHz, Q:20, Idc:1,1A, Rdc:0.07Ω
- Condensador electrolítico de aluminio RS Pro, 2200µF, ±20%, 16 V dc, Serie RS
- 1 Zócalo de 40 pines (Microprocesador).
- 3 Zócalos de 8 pines (Operacionales y conversor DC/DC 5v a -4,75).
- 1 Zócalo de 16 pines (Convertidor digital analógico).
- 1 Oscilador de 3 MHz.
- 1 Conector de alimentación Mj-179PH
- Diodo Zener de 12v (se utiliza en el conversor)

3.3 Materiales acondicionadores

Estos materiales hacen de soporte para una optimización del dispositivo, físicamente.

3.3.1 Pota cubetas y porta leds

El porta cubetas es la pieza encargada de alojar la muestra a medir. Tiene unas medidas de 50x25x40mm y es de acero. Está preparada para cubetas estándar de 10mm. Tiene un orificio alineado con la fuente de luz para alojar el fotodiodo encargado de recoger la señal de salida de la muestra en la medida de la absorbancia. También cuenta otro orificio en perpendicular al haz de luz para la medida de la fosforescencia.

El porta leds es una lámina de acero de 50x50x5mm y es la encargada de mantener los LED alineados correctamente. Tiene tres agujeros de 8mm de diámetro para poder introducir las monturas en cuyo interior se colocan los diodos LED. En este caso solo se utilizara el orificio intermedio donde se ubicara

el led tricolor y los otros orificios serán tapados con cinta aislante para no contaminar la toma de medidas. En la figura 17 se puede ver la estructura de dicho elemento.

Figura 17: Porta cubetas y porta leds.

3.3.2 Tacos

Son unos tacos de goma que se pegan en la placa base del dispositivo para aportar una mayor consistencia.

3.3.3 Material software

Para el desarrollo de este proyecto se han utilizado los siguientes programas:

3.3.4 MPLAB

Es un entorno de desarrollo integrado (IDE), que posee un conjunto de herramientas libres e integradas para el desarrollo de aplicaciones integradas que emplean microchips PIC y dsPIC microcontroladores. Que a través del copilador HI-TECH nos permite programar en lenguaje C.

3.3.5 Matlab

Es un el lenguaje de alto nivel y entorno interactivo, utilizado por millones de ingenieros y científicos en todo el mundo. Le permite explorar y visualizar ideas, así como colaborar interdisciplinarmente en procesamiento de señales e imagen, comunicaciones, sistemas de control y finanzas computacionales. De este programa se ha utilizado la herramienta prtool.

3.3.6 DesignSpark PCB 6.0

Es un software gratuito para diseñar pcb, se encuentra libre de limitaciones como tamaño de la plaqueta, numero de pines o número de capas. No se queda solo en diseñar pcb sino que también puede crear esquemáticos y obtener como salida archivos Gerber, para poder elegir libremente quien nos fabricara la placa (o nosotros mismos si tenemos un equipo CNC).

Capítulo 4: Desarrollo de la Programación

En este apartado se determinara el rendimiento de la red en función del número de neuronas de las distintas capas y el tamaño que ocupa la memoria de datos. También se estudiará el código que utiliza Matlab para generar las salidas de la red neuronal y finalmente se realiza la programación del código en C para las distintas versiones.

4.1 Trabajo con *nprtool* de Matlab

En este apartado se van a describir las tareas a realizar con la herramienta de Matlab.

4.1.1 Obtención de bases de datos

Para poder trabajar con la herramienta nprtool de Matlab hay que realizar una base de datos previa. Esta base de datos se consigue a partir de la medición de muestras con un prototipo inicial de espectrofotometría con el cual se caracterizan las distintas muestras para tres parámetros de entrada que se obtienen a partir de la medición a diferentes longitudes de onda. A partir de estos se determinan las salidas que clasificarán el tipo de muestra según:

- Caso 1: La concentración es roja o azul.
- Caso 2: Si la concentración es mayor del 50% o no.
- Caso 3: Si la concentración es del 25%,50%,75% y 100%.

También se realizará una base de datos posterior siguiendo la misma metodología que la previa pero utilizando el prototipo mejorado de espectrometría. Dicha base de datos se adjuntan en el apartado de anexos.

4.1.2 Estudio que determina el rendimiento de la red en función del número de neuronas de las distintas capas y el tamaño que ocupa de la memoria de datos

Una vez obtenidos los tres casos de bases de datos se procede al entrenamiento de la red neuronal, para ello se accede a la herramienta de Matlab nprtool y se inserta la base de datos en función del caso de estudio. También se ira variando el número de neuronas de la capa oculta y se observa el rendimiento de las clasificaciones. Dicho estudio está presente en el apartado de anexos. Tras las conclusiones se puede afirmar que:

- En el caso de 2 neuronas de salida se trabajara en el rango de 2 a 5 neuronas ocultas.
- En el caso de 4 neuronas de salida se trabajara con 3 y 10 neuronas ocultas.

- En el caso de 8 neuronas de salida se trabajara con 4, 8 y 9 neuronas ocultas.

4.1.3 Estudio del código que utiliza Matlab para generar las salidas de la red neuronal

Con este estudio se es capaz de ver el funcionamiento de la red cuando ya han sido ajustados todos los parámetros de cada capa el cuál se encuentra en los anexos. En el diagrama de bloques de la figura 18 se explica su funcionamiento:

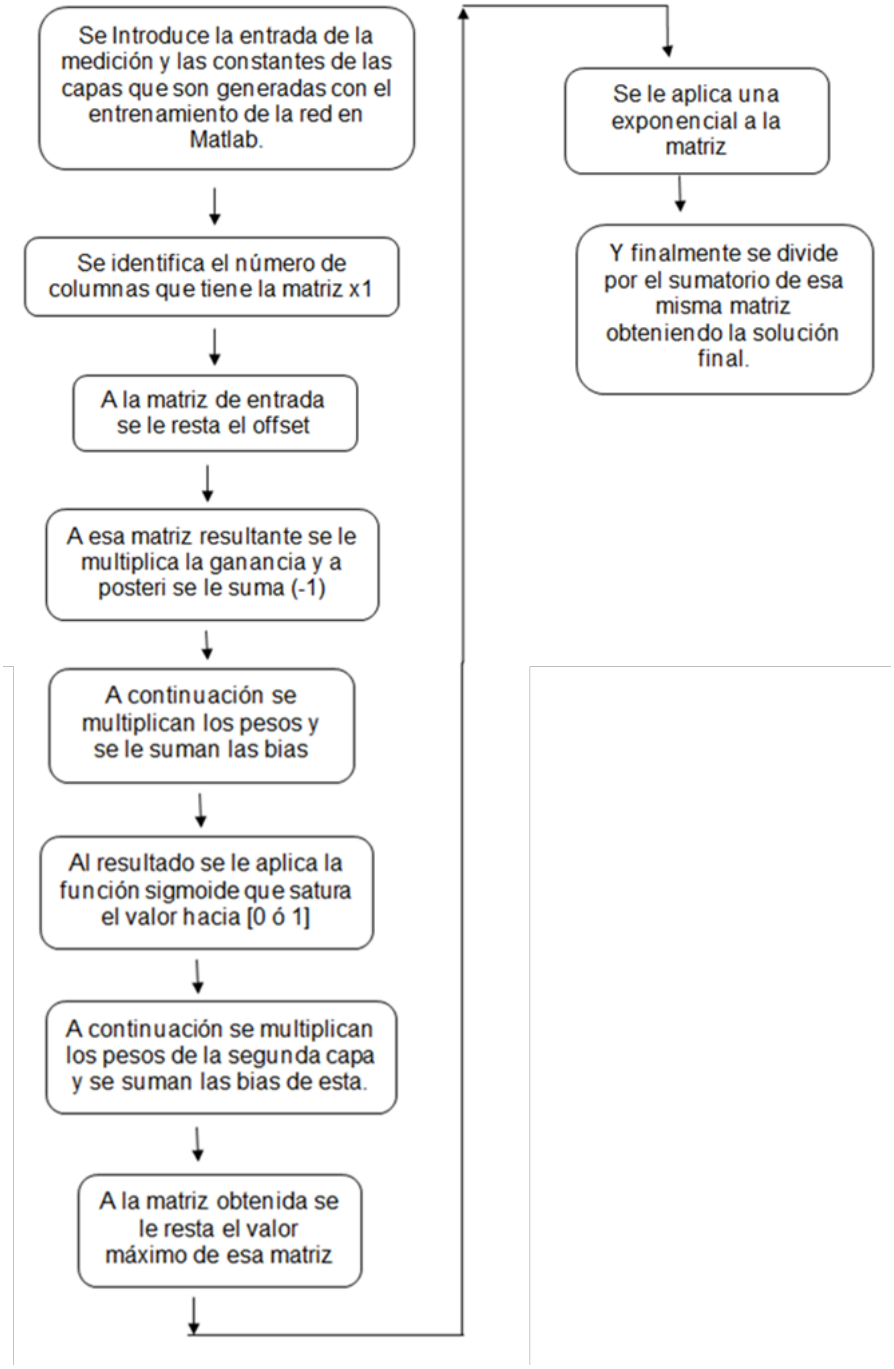


Figura 18: Diagrama de flujo del código generado con Matlab

4.2 Programación en C:

En el siguiente apartado se van a explicar las distintas versiones del programa en forma de diagrama de flujo y los requisitos de memoria reales alcanzados. Además este código posee un desarrollo matemático que se encuentra en la hoja de anexos.

4.2.1 Versión1: 3 neuronas de entrada, 1 oculta y 2 de salida.

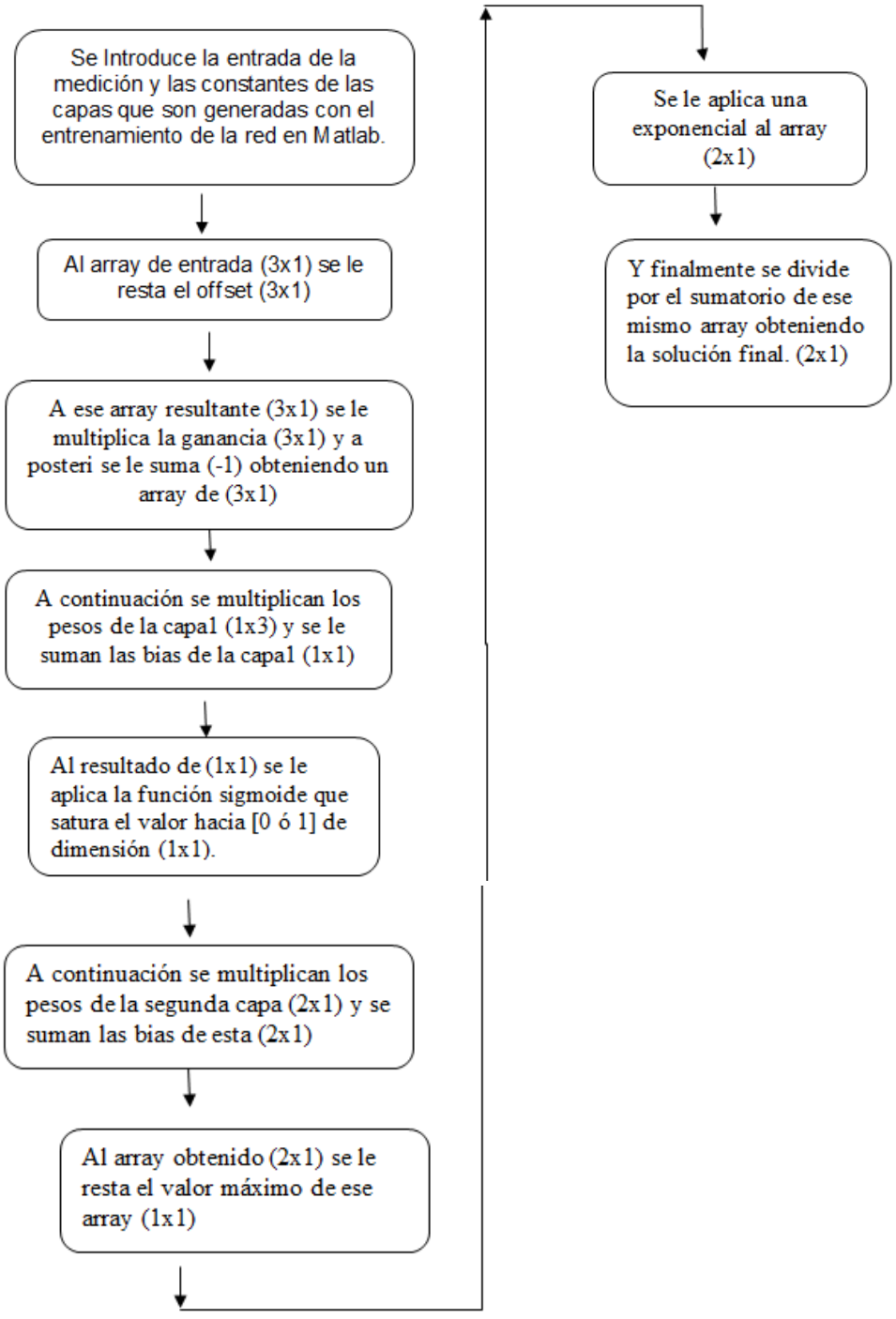


Figura 19: Diagrama de flujo

```
Memory Summary:
    Program space        used   E2Fh (  3631) of  2000h words   ( 44.3%)
    Data space           used    98h (   152) of   170h bytes   ( 41.3%)
    EEPROM space         used     0h (     0) of   100h bytes   (  0.0%)
    Configuration bits   used     1h (     1) of     1h word    (100.0%)
    ID Location space    used     0h (     0) of     4h bytes   (  0.0%)
```

Figura 20: Resumen de la capacidad de memoria del PIC

4.2.2 Versión 2: 3 neuronas de entrada, 1 oculta y n2 de salida

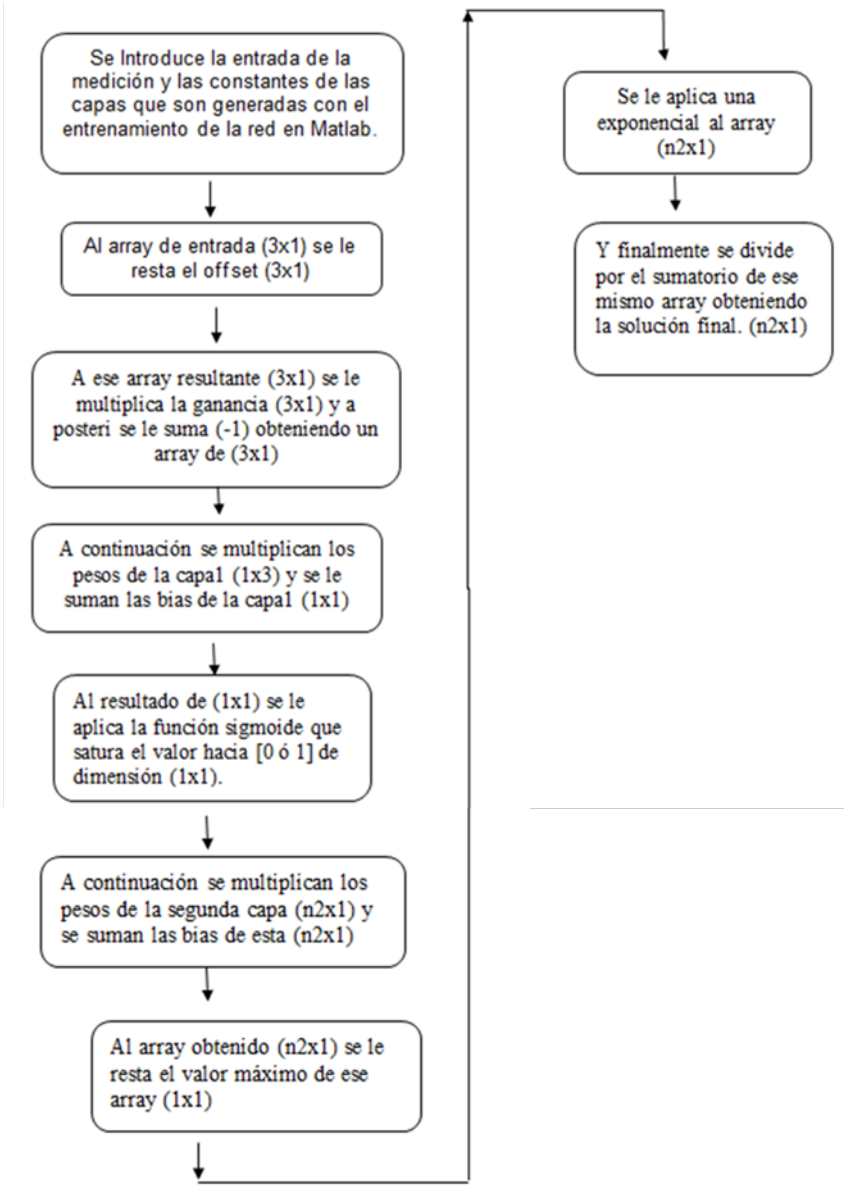


Figura 21: Diagrama de flujo

```
Memory Summary:
    Program space        used   EF8h (  3832) of  2000h words   ( 46.8%)
    Data space           used    71h (   113) of   170h bytes   ( 30.7%)
    EEPROM space         used     0h (     0) of   100h bytes   (  0.0%)
    Configuration bits   used     1h (     1) of     1h word    (100.0%)
    ID Location space    used     0h (     0) of     4h bytes   (  0.0%)
```

Figura 22: Resumen de la capacidad de memoria del PIC

4.2.3 Versión 3: n0 neuronas de entrada, n1 ocultas y n2 salida

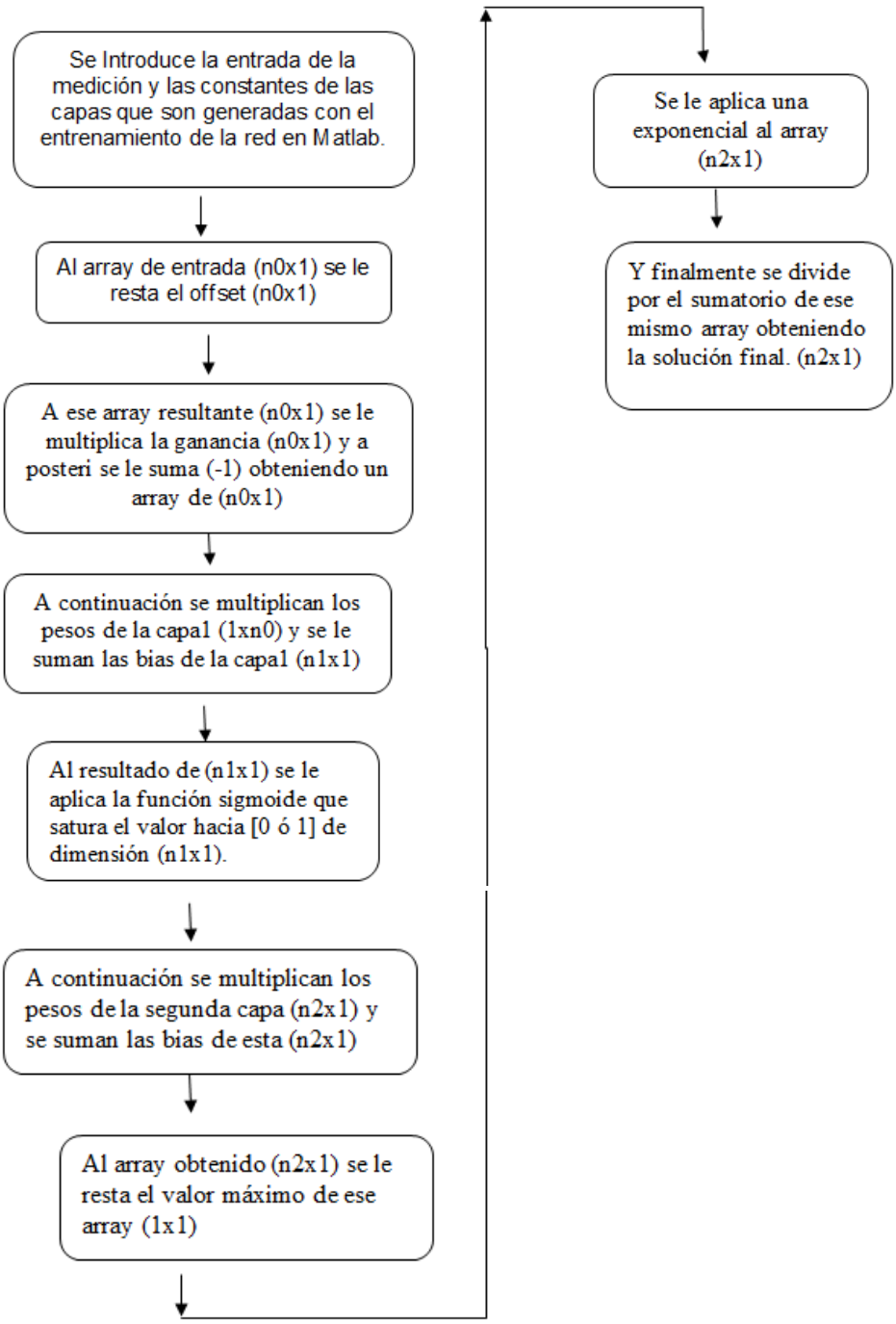


Figura 23: Diagrama de flujo

```
Memory Summary:
    Program space        used   FA8h (  4008) of  2000h words   ( 48.9%)
    Data space           used    80h (   128) of   170h bytes   ( 34.8%)
    EEPROM space         used     0h (     0) of   100h bytes   (  0.0%)
    Configuration bits   used     1h (     1) of     1h word    (100.0%)
    ID Location space    used     0h (     0) of     4h bytes   (  0.0%)
```

Figura 24: Resumen de la capacidad de memoria del PIC

4.3 Estudio de la capacidad de memoria y el tiempo de ejecución de la Red Neuronal en función del número de neuronas de la capa oculta y salida.

En primer lugar se calcula con la ayuda del compilador de MPLAB la memoria de programa y la memoria de datos de la función red neuronal general y posteriormente el tiempo máquina en función del número de neuronas de la capa oculta y salida.

Para poder llevar a cabo este estudio es necesario ir variando las constantes de la reden las distintas capas, para ello se utiliza la herramienta nprtool de Matlab que nos permite obtener estas constantes de entrenamiento. En la tabla 3 quedan recogidos los resultados obtenidos del proceso descrito.

Nº neuronas oculta	Nº neuronas salida	Memoria de Programa Red (posiciones)	Memoria de datos Red (posiciones)	Tiempo Máquina (Cycles)	Tiempo de ciclo (msecs)
1	2	4184	159	76413	76413000
3	2	4230	165	77593	77593000
5	2	4394	174	233051	233051000
1	4	4196	165	114233	114233000
3	4	4254	171	224886	224886000
5	4	4433	179	307088	307088000
1	8	4229	177	258082	258082000
3	8	4433	185	366149	366149000
5	8	4505	191	471220	471220000

Tabla 3

A continuación se calcula la memoria de programa del Menú y las medidas, para ello se compilan el código total y posteriormente la Red:

$$Memoria\ código\ Total - Memoria\ código\ Red = Memoria\ menu\ y\ medida$$

Ecuación 4

En la tabla 4 se pueden ver los resultados obtenidos en cuanto a memoria de programa y datos del menú de ventanas y medidas.

Memoria de programa entero (bytes)	Memoria de Datos entero (bytes)	Memoria de programa Red (bytes)	Memoria de datos Red (bytes)	Memoria programa Menú y medida (bytes)	Memoria datos Menu y medidas (bytes)
6976	181	4394	174	2582 (cte)	7 (cte)

Tabla 4

Un dato interesante de cálculo es el margen de la memoria de programa y de datos que tiene la Red Neuronal para cumplir con los requisitos de memoria exigidos por el PIC, para ello se realiza el siguiente cálculo:

$$Memoria\ Total\ PIC - Memoria\ programa\ Menu\ y\ medida = Margen\ total\ de\ la\ Red$$

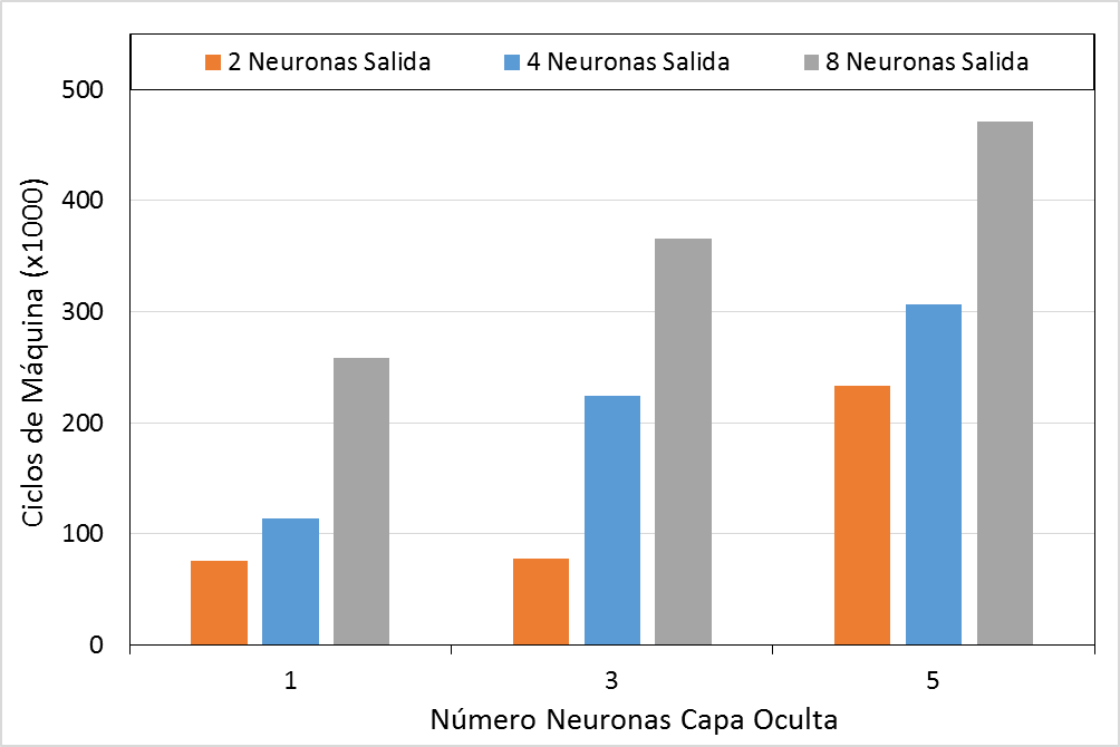
Ecuación 5

El resultado de la Ecuación 1 se muestra en la tabla 5. Al observar la tabla 3 se puede concluir que en todos los casos estudiados se cumplen los requisitos de memoria del PIC

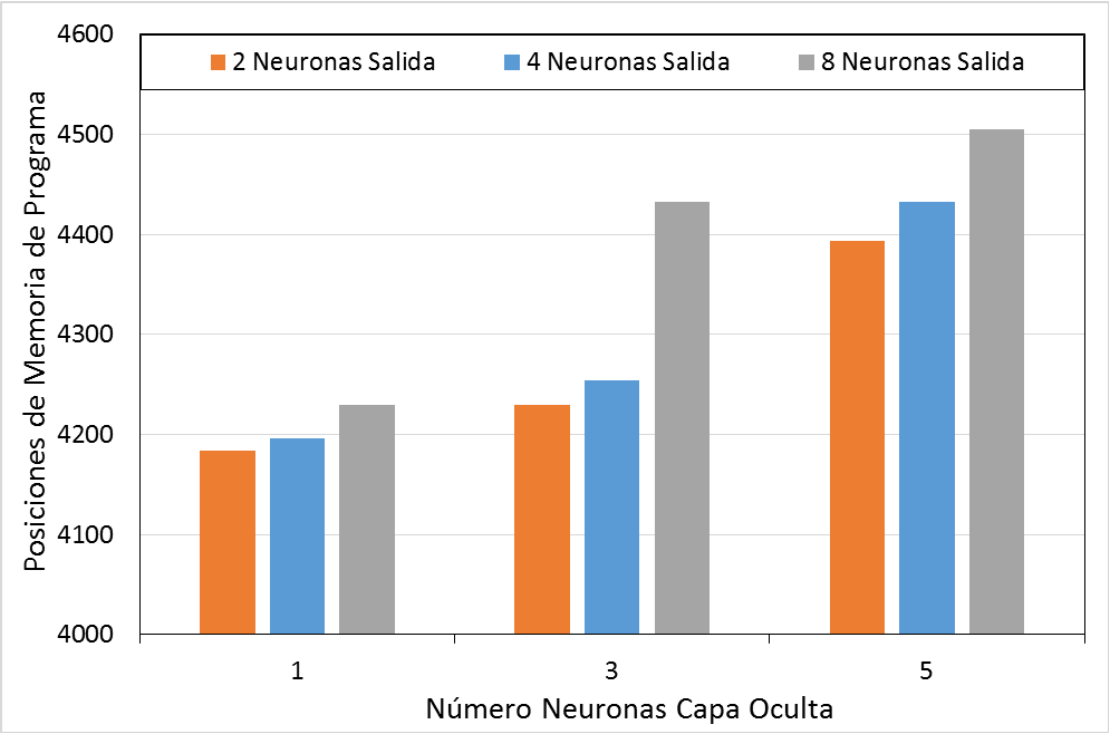
Margen Memoria programa Red	Margen memoria datos Red
5610 posiciones	361 posiciones

Tabla 5

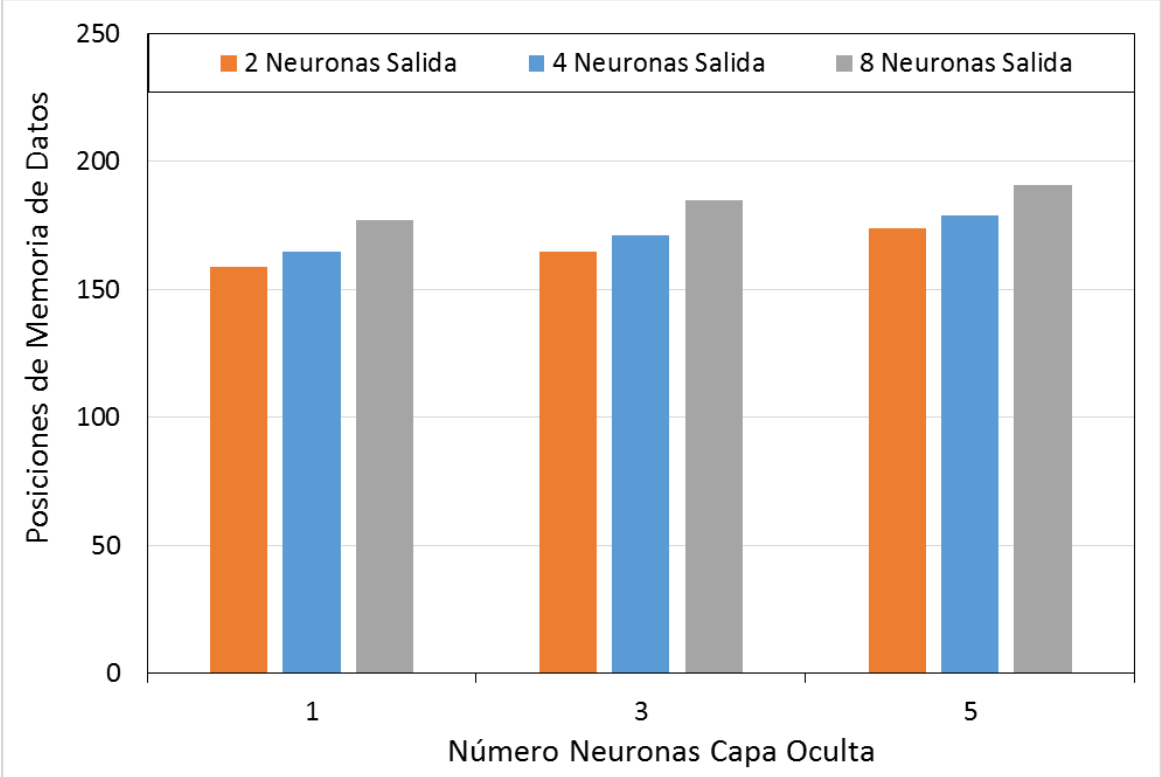
Para Terminar este estudio se muestran de manera gráfica la evolución de la memoria de datos, la memoria de programa y el ciclo máquina en función del número de neuronas de la capa oculta para las series de 2,4 y 8 neuronas de salida.



Gráfica 1



Gráfica 2



Gráfica 3

La Conclusión a estos resultados es que a medida que se aumenta el número de neuronas de la capa oculta, se incrementan los bytes de la memoria de datos, la memoria de programa y los ciclos del tiempo máquina para las distintas neuronas de salida.

También se aprecia que a mayor número de neuronas de la capa de salida tomando contante las neuronas ocultas, el tiempo máquina, la memoria de datos y la memoria de programa es mayor.

Además se puede observar cómo se dispara el tiempo máquina para 5 neuronas ocultas para las respectivas neuronas de salida con respecto a las neuronas ocultas de 1 y 3.

En la memoria de programa se puede ver como aumenta de manera notable para el caso de 8 neuronas de la capa de salida y 5 neuronas en la capa oculta. También se dispara cuando se pasa de 3 a 5 neuronas ocultas para los distintos casos de neuronas de salida.

Para terminar la memoria de datos no sufre variación entre las distintas neuronas ocultas para las distintas neuronas de salida.

Capítulo 5: Mejora Electrónica del Prototipo

En este apartado se va a describir el funcionamiento, las modificaciones que ha sufrido el prototipo inicial y los cálculos necesarios para el ajuste de los respectivos elementos. También se aportarán esquemas para una mejor visualización.

5.1 Elementos eliminados que no se van a utilizar

En la figura 25 se pueden ver la eliminación de la parte de sensado de la fosforescencia y la configuración de encendido de los leds.

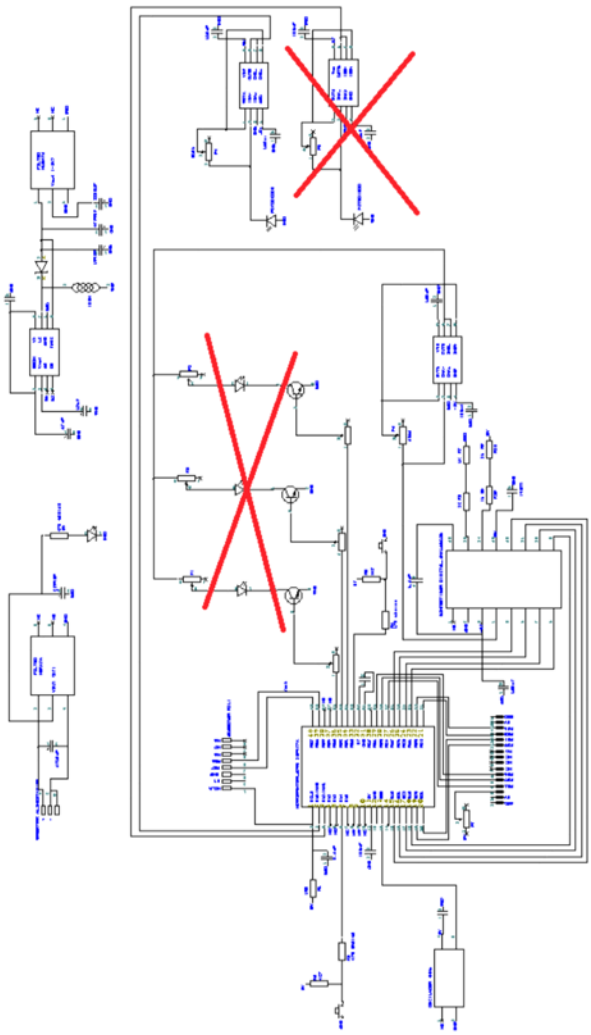


Figura 25: Esquemático del prototipo inicial

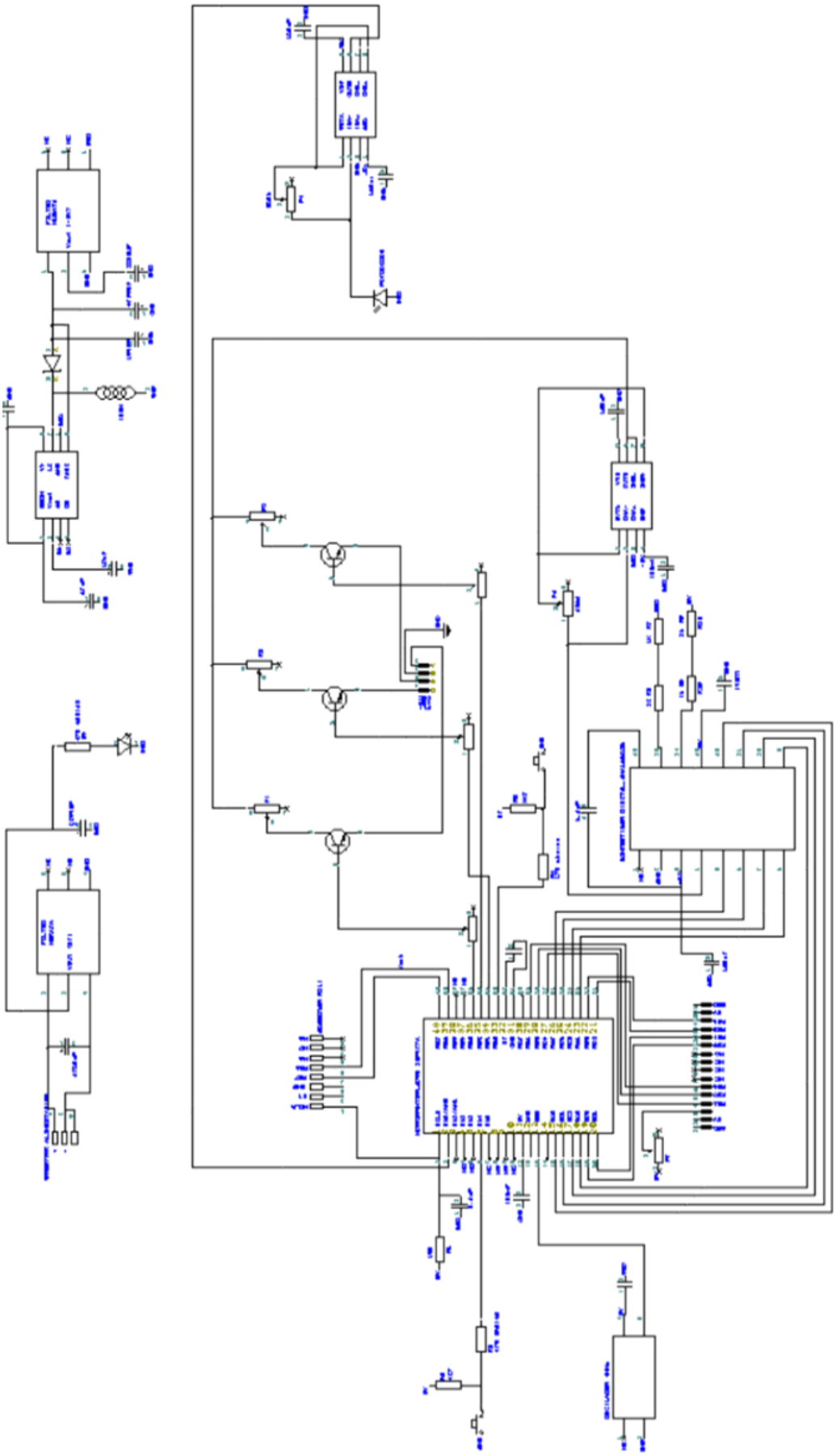


Figura 26: Esquemático del prototipo final

En la figura 25 se observa cómo queda el esquema general del circuito final y como están conectadas las distintas partes entre sí. En todas las alimentaciones de los integrados se han colocado condensadores de desacoplo de 100nF. Para poder explicar mejor el circuito se ha dividido en bloques donde se explica más detalladamente.

5.2 Bloque de diodos leds:

Este bloque es el encargado de producir la luz de diferentes longitudes de onda que se hacen pasar a través de la muestra.

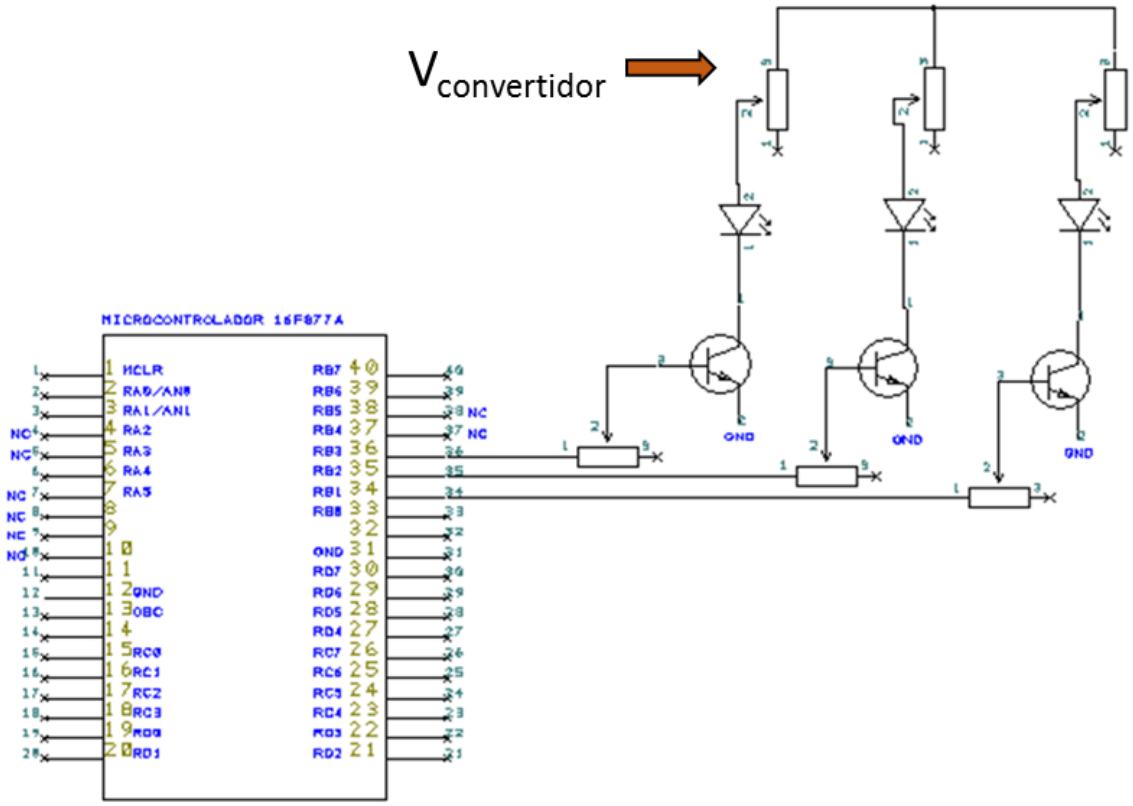


Figura 27: bloque de diodos leds del prototipo inicial

Este circuito de la figura 27 ha sido modificado para poder colocar un diodo tricolor y así ser mucho más preciso a la hora de tomar las mediciones de absorbancia, ya que la posición del foco emisor es constante para las tres longitudes de onda y la altura donde se encuentra el fotodiodo receptor es la misma. El circuito final se puede ver en la figura 28.

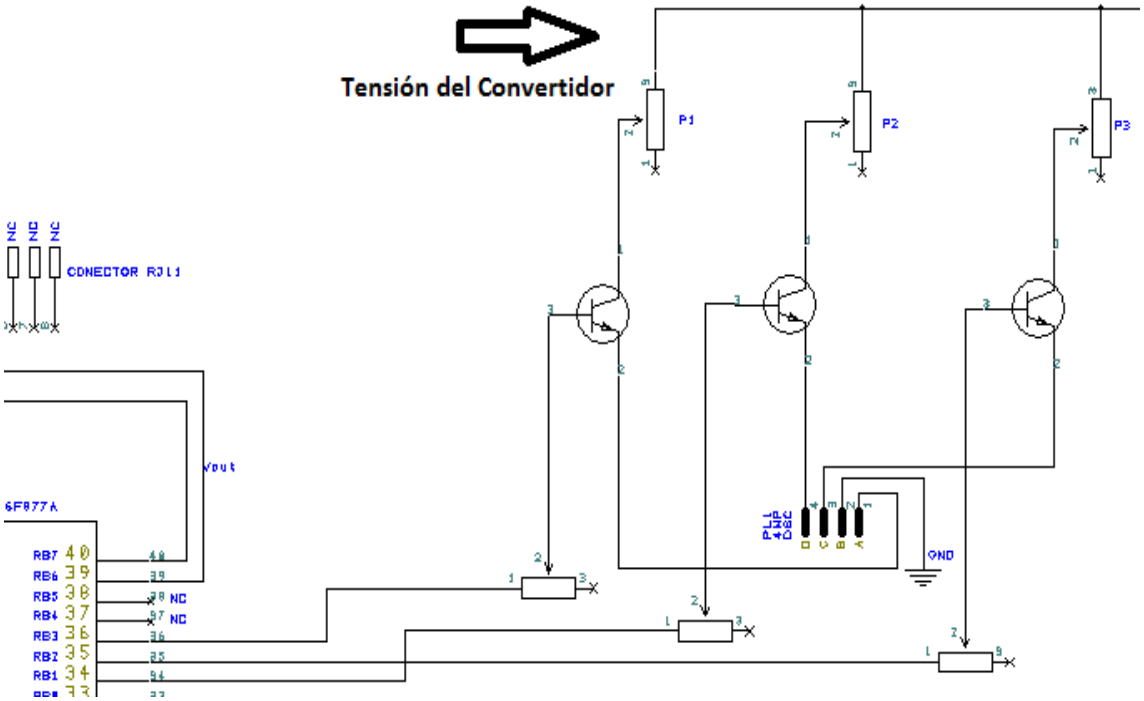


Figura 28: Bloque de led tricolor del prototipo final

Es un sistema sencillo para el encendido y apagado del diodo tricolor. Los transistores, BC548, son controlados por las salidas RB1, RB2 y RB3 del microcontrolador. Estas salidas se ponen a 1 cuando se le da la orden mediante los pulsadores y la pantalla LCD. El programa del microcontrolador se encarga de llevar la secuencia, el tiempo de encendido y apagado y la potencia a cada una de las patillas del led tricolor.

Las salidas están conectadas a la base de los transistores mediante una resistencia variable. Cuando las salidas están a uno, en la patilla correspondiente hay un voltaje de 5v, y cuando está a cero en la patilla hay 0v. De este modo cuando hay un uno, el transistor se satura y el LED se ilumina. El voltaje que le llega al LED viene del convertido y es configurable.

Los cálculos se han hecho con el voltaje máximo que da el convertidor que será cuando pase la mayor intensidad por el colector del transistor.

5.2.1 Cálculo de las Resistencia de colector de las tres ramas de leds:

En todos los casos se han colocado potenciómetros cercanos al valor calculado para poder ajustar manualmente y con más precisión la intensidad.

$$Resistencia_{Colector} = \frac{V_{convertidor} - V_{diodo} - V_{CE}}{I_{colector}}$$

Ecuación 6

Led Hyper Red:

$$R_{colector} = \frac{5 - 1{,}95 - 0{,}2}{20\ mA} = 142{,}5\ \Omega$$

Ecuación 7

Led Blue:

$$R_{Colector} = \frac{5 - 3{,}3 - 0{,}2}{20\ mA} = 75\ \Omega$$

Ecuación 8

Led Green:

$$R_{Colector} = \frac{5 - 3{,}3 - 0{,}2}{20\ mA} = 75\ \Omega$$

Ecuación 9

5.2.2 Cálculo de la corriente de base:

Esta es la intensidad mínima de base para que se sature el transistor.

$$I_{base} = \frac{I_{colector}}{\beta} = \frac{20\ mA}{150} = 133{,}33\ \mu A$$

Ecuación 10

5.2.3 Cálculo de las Resistencias de base de las tres ramas.

La intensidad de base para que se sature el transistor deberá ser igual o mayor a la calculada.

$$R_{Base} = \frac{V_{convertidor} - V_{BE} - V_{diodo}}{I_{base}}$$

Ecuación 11

Led Hyper Red:

$$R_{Base} = \frac{5 - 0{,}7 - 1{,}95}{133{,}33\ \mu A} = 17{,}625\ K\Omega$$

Ecuación 12

Led Bue:

$$R_{Base} = \frac{5 - 0{,}7 - 3{,}3}{133{,}33\ \mu A} = 7{,}5\ K\Omega$$

Ecuación 13

Led Green:

$$R_{Base} = \frac{5 - 0{,}7 - 3{,}3}{133{,}33\ \mu A} = 7{,}5\ K\Omega$$

Ecuación 14

Con estos cálculos se garantiza que los transistores se saturen para el led tricolor. Si se desea cambiar este led por otro diferente habrá que calcular la resistencia de colector necesaria para que circule la corriente nominal. Después de esto se ajustar manualmente el potenciómetro correspondiente para que por el colector circule la intensidad deseada.

5.3 Bloque del conversor digital analógico:

Con este bloque y mediante el menú, se puede seleccionar la luminosidad de los diodos LED. Habiendo calculado el circuito anterior para un voltaje de alimentación máximo de 5v, si se disminuye el voltaje de alimentación disminuye la intensidad que pasa por el LED y por lo tanto su luminosidad. El circuito estándar para su aplicación más típica se muestra en la figura 29.

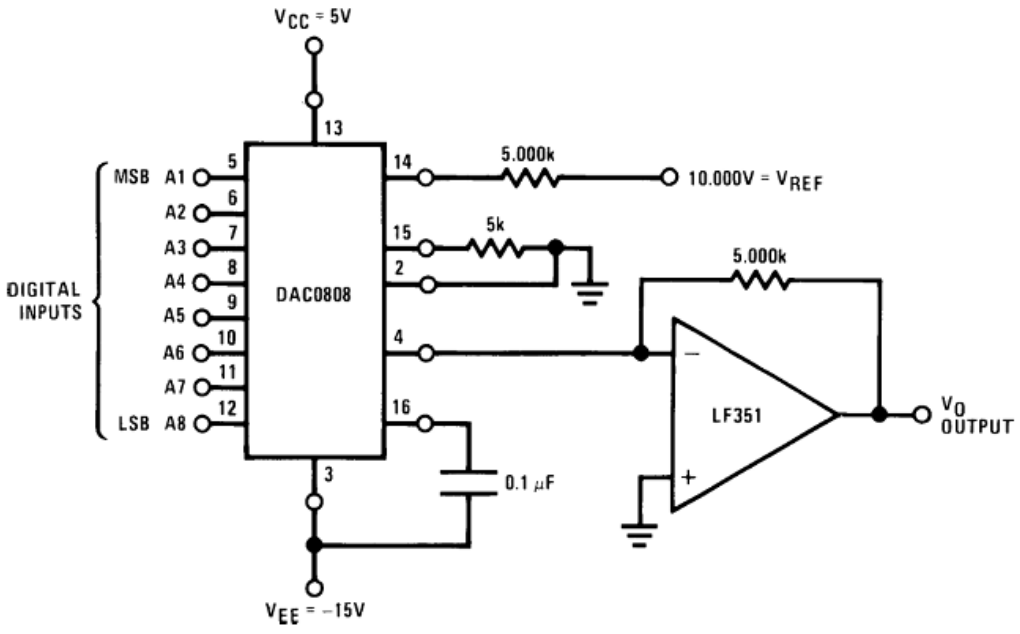


Figura 29: Circuito DAC 0808

El integrado del convertidor DAC 0808 está alimentado entre 5v y -5v en las patillas Vcc y Vee, que son las patillas 13 y 3 respectivamente. Los voltajes

de referencia y las resistencias son a modo de ejemplo, dependiendo del valor que queramos conseguir a la salida estos variarán.

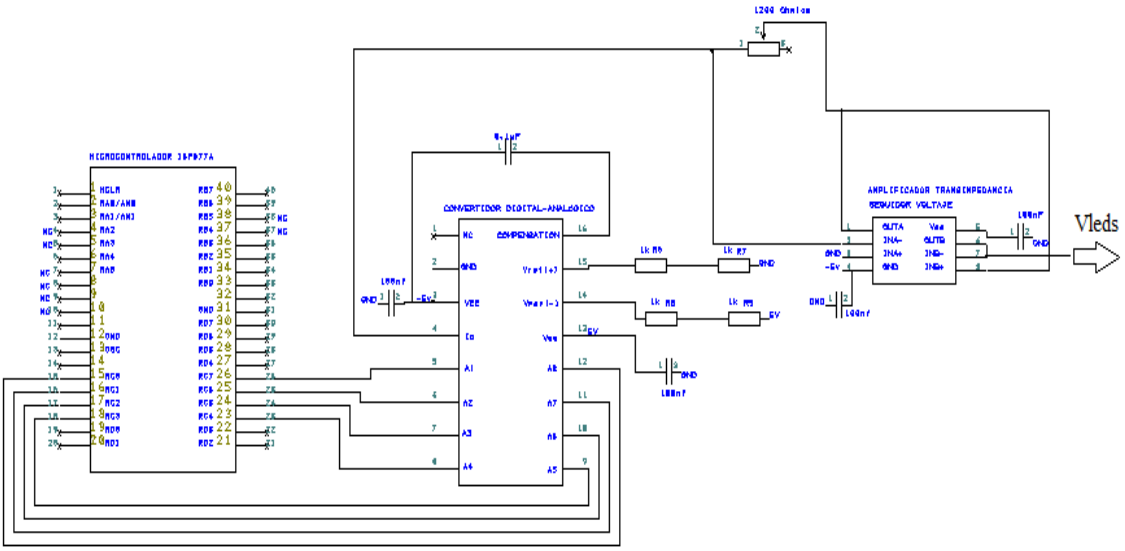


Figura 30: Bloque conversor analógico-digital

El conversor está conectado al puerto C del microcontrolador por el cual sale el número binario correspondiente al voltaje que se desea sacar. Este número entra al puerto paralelo del conversor correspondientes a las patillas de la 5, bit más significativo, hasta la 8, bit menos significativo.

La intensidad de salida viene dada por la siguiente fórmula:

$$Io = \left(\frac{Vref}{R6+R7}\right) * \left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A6}{62} + \frac{A7}{128} + \frac{A8}{256}\right)$$

Ecuación 15

Las resistencias R6 y R7 se han escogido de manera arbitraria ya que en lo único que influyen es en el rango de la corriente de salida, que en función de la resistencia que se pone en el amplificador de transimpedancia posterior da siempre lo mismo. Para la mayor corriente de salida tendrán que corresponder 5v y para la menor corriente 0v. Con esto y escogiendo 2k como resistencia, 5v de Vref+ y 0v de vref- tendremos:

$$Iomax = \frac{5v}{2000} * \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256}\right) = 2.49\ mA$$

Ecuación 16

Esta intensidad entra en el integrado MCP6282, que contiene dos amplificadores operacionales como se puede apreciar en la figura 31.

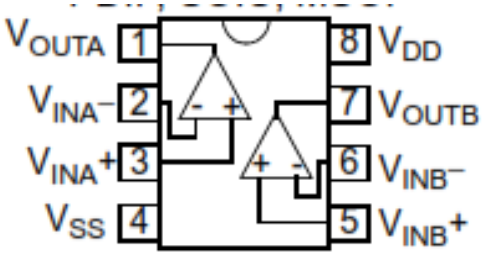


Figura 31: Esquema del MCP6292

Así si tenemos que la corriente máxima de salida son 2.49 mA y se quieren conseguir 5v la resistencia del amplificador deberá ser:

$$R = \frac{5v}{2.49mA} = 2008\ ohmios$$

Ecuación 17

Este voltaje se introduce en el siguiente amplificador del integrado en montaje de seguidor de tensión para que el circuito de los diodos LED esté aislado con respecto al bloque del convertidor.

El conversor tiene una resolución de 8 bit, por lo tanto cada bit variará el voltaje a la salida según la ecuación 18.

$$= \frac{5v}{2^8} = 0.0195\ v$$

Ecuación 18: Resolución del convertidor

5.4 BLOQUE RECOGIDA Y ACONDICIONAMIENTO DE SEÑAL

En este bloque y mediante el fotodiodo, la luz que atraviesa la muestra es recogida y tratada para poder interpretarla. Se disponen de dos circuitos iguales, uno para la medida de la transmitancia y absorbancia, en el cuál el fotodiodo se encuentra enfrentado a la fuente de luz. El segundo es para la medida de la fosforescencia, en el cual el fotodiodo se encuentra perpendicular a la fuente de luz para que solo capte la luz emitida por la muestra y no la de la fuente.

En el prototipo final se ha eliminado el circuito de la fosforescencia y solo se utilizará el circuito de la absorbancia como se puede ver en la figura 32. Para este se ha utilizado el mismo integrado que en el bloque anterior y en la misma disposición. El fotodiodo crea una corriente según la luz que incide en él y en la dirección cátodo-ánodo. Esta intensidad se convierte en voltaje mediante el amplificador de transimpedancia, ya que a continuación se introduce en el micro para su interpretación. Seguidamente se coloca un seguidor de tensión para aislar los diferentes circuitos.

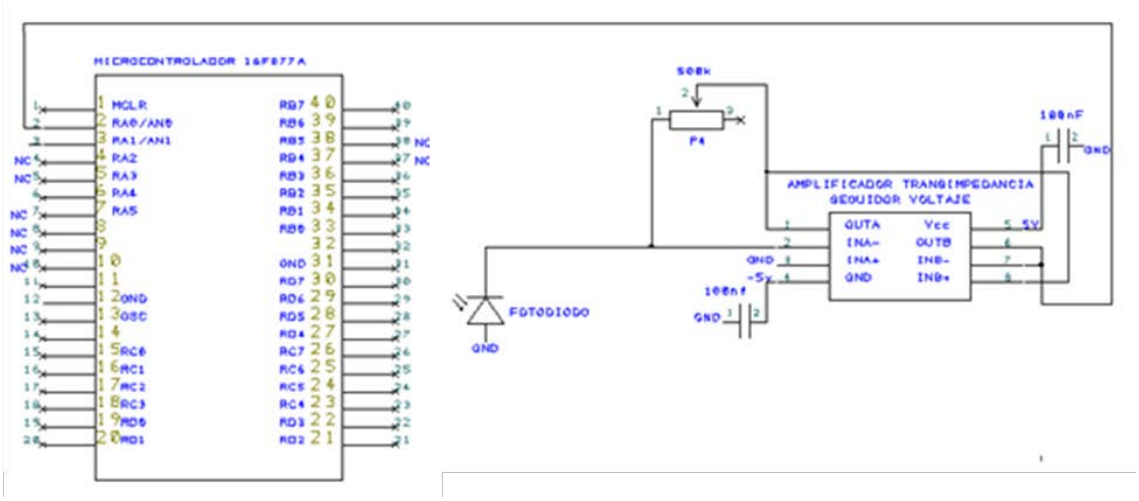


Figura 32: Bloque de recogida y acondicionamiento de la señal del prototipo final.

Para el cálculo de la resistencia del amplificador, se han hecho una serie de pruebas para ver cuál es el diodo que más luz emite y por lo tanto más corriente genera. En este caso el diodo más luminoso es el de 626 nm que produce una intensidad en el fotodiodo de 92uA aproximadamente. Si queremos adaptar el máximo rango de entrada del convertidor analógico-digital del micro, el cuál son 5v, se debe tomar esta señal como la máxima y por lo tanto deberá producir 5v aproximadamente. Para ello la resistencia deberá ser:

$$R = \frac{5v}{92\mu A} = 54347.82\ \Omega$$

Ecuación 19

En las pruebas los otros diodos LED han dado una señal aproximada de 6µA, con lo que a la salida dan una tensión de:

$$Vout = 6\mu A * 54347.82 = 0.326\ v$$

Ecuación 20

Esta tensión quizá sea un poco pequeña y la poca variación de intensidad producida por estos diodos LED producirá poca variación de voltaje, lo cuál puede ser un problema. Para solucionar esto lo que se ha hecho es tomar como máxima señal la producida por el LED de 626 nm cuando está alimentado a 2.5v, el cuál produce una señal de 22µA. Con esto la resistencia a implementar será:

$$R = \frac{5v}{22\mu A} = 227272.72\ \Omega$$

Ecuación 21

Para poder ajustar la escala al gusto del usuario, y por si en un futuro se colocan diferentes diodos LED con diferentes respuestas, se ha colocado un potenciómetro de 500k para hacer el ajuste manual.

5.5 BLOQUE PULSADORES, OSCILADOR Y CONECTOR RJ11

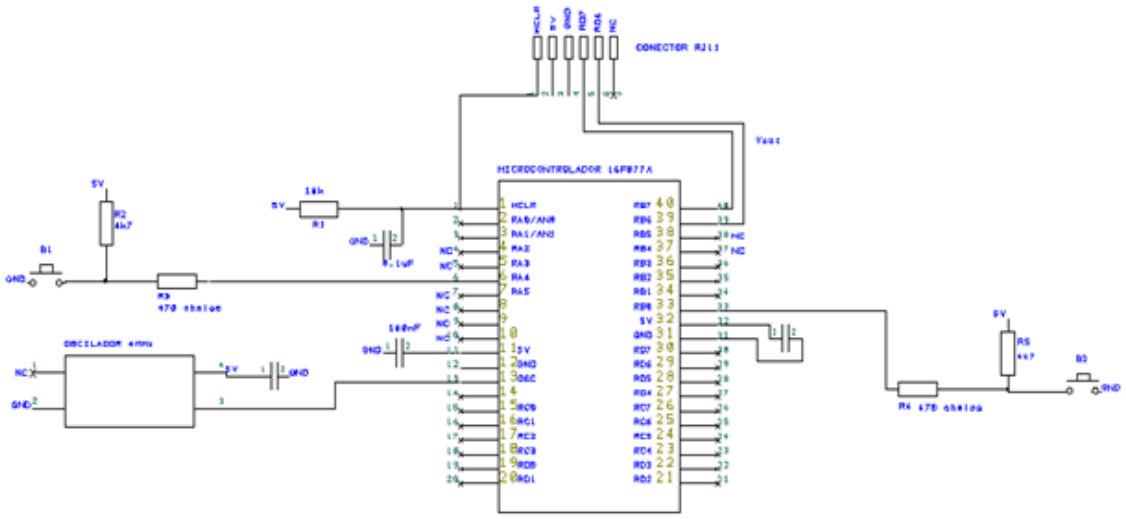


Figura 33: Bloque pulsadores, oscilador y conector RJ11.

Para poder movernos por el menú, se han incorporado los pulsadores B1 y B2 conectados a las entradas del micro RA4 y RB0 respectivamente. Los pulsadores están conectados con una lógica inversa, es decir cuando no están pulsados, en la entrada se obtiene un 1 y cuando se pulsan en la entrada se obtiene un 0.

El circuito cuenta con un oscilador externo de 4 MHz conectado a la patilla 13 del micro. El microcontrolador cuenta con un oscilador interno de 20 MHz pero se incorpora la opción de trabajar a diferentes frecuencias.

Para poder programar el microcontrolador sin necesidad de ser desalojado del circuito se ha colocado un conector RJ11. Este conector está configurado para ser conectado a un debugger ICD2, las patillas de este están conectadas de la siguiente manera:

- Patilla 1: MCLR (patilla 1 micro)
- Patilla 2: 5v
- Patilla 3: GND
- Patilla 4: PGD (patilla 40 micro)
- Patilla 5: PGC (patilla 39 micro)
- Patilla 6: NC

5.6 BLOQUE PANTALLA LCD

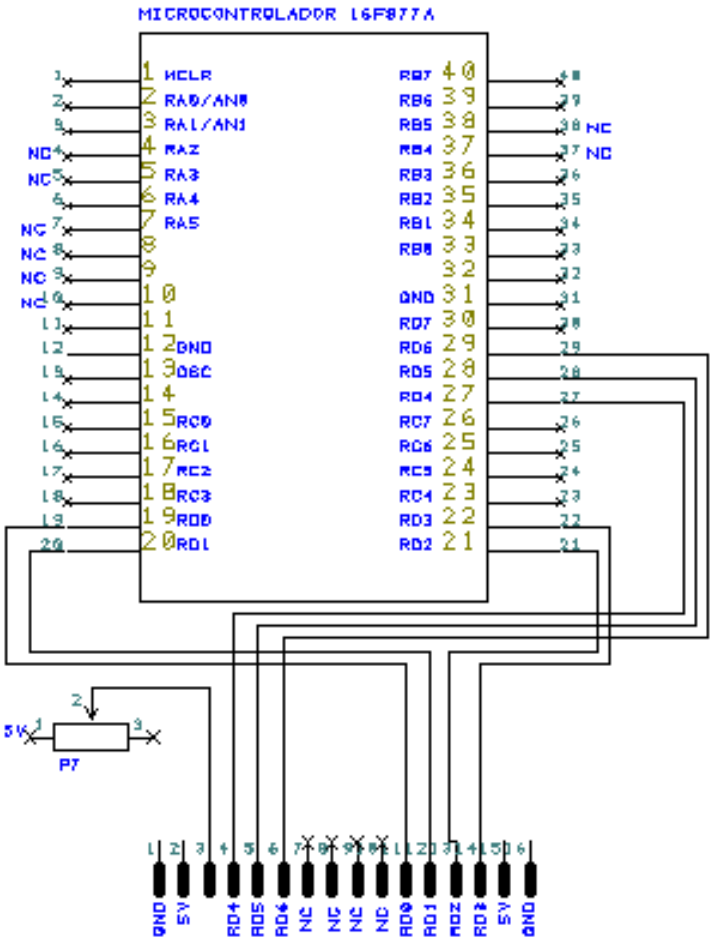


Figura 34: Bloque pantalla LCD

La pantalla LCD está controlada mediante las salidas del puerto B del microcontrolador. Está configurada para trabajar en módulo de 4bit, es decir los datos se envían por un bus de 4 pines. Las patillas de la pantalla LCD están conectadas de la siguiente manera:

- Pin 1: GND
- Pin 2: 5v
- Pin 3: potenciómetro 10k. Ajuste de contraste.
- Pin 4: RD4 (patilla 27 micro). Para diferenciar si llega un dato o una instrucción.
- Pin 5: RD5 (patilla 28 micro). Para leer o escribir.
- Pin 6: RD6 (patilla 29 micro). Señal de habilitación del módulo LCD.
- Pin 7-8-9-10: No conectados
- Pin 11-12-13-14: RD0, RD1, RD2, RD3 (patillas19, 20, 21,22 micro).Bus de datos.
- Pin 15: 5v. Ánodo luz LED
- Pin 16: GND .Cátodo luz LED

5.7 Bloque de Alimentación:

Para adaptar la tensión de la red a las necesidades del circuito se incorporan varias etapas en el circuito. Primeramente, la fuente de alimentación adapta la señal alterna de la red a 5 V_{DC}. Después se implementa una etapa que consta de dos filtros de choke y un conversor DC/DC de 5v a -5v.

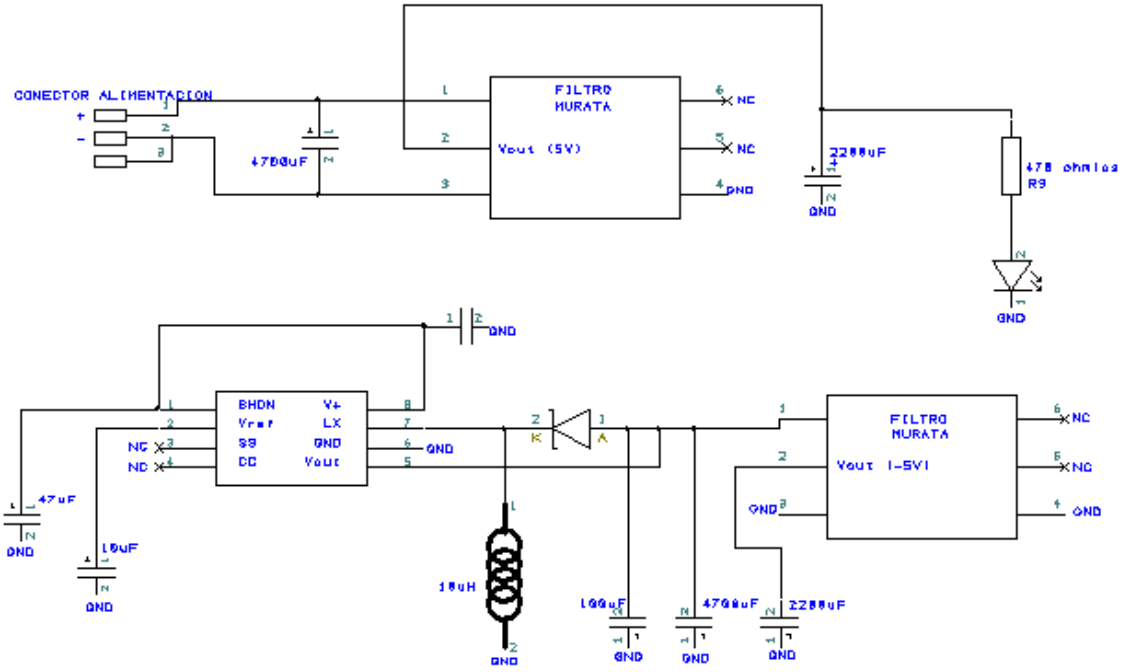


Figura 35: Bloque alimentación

Para eliminar posibles ruidos de la red y de la propia fuente de alimentación, se ha colocado un filtro Murata BNX002 en cuyo interior hay un filtro LC en la disposición de la figura 36.

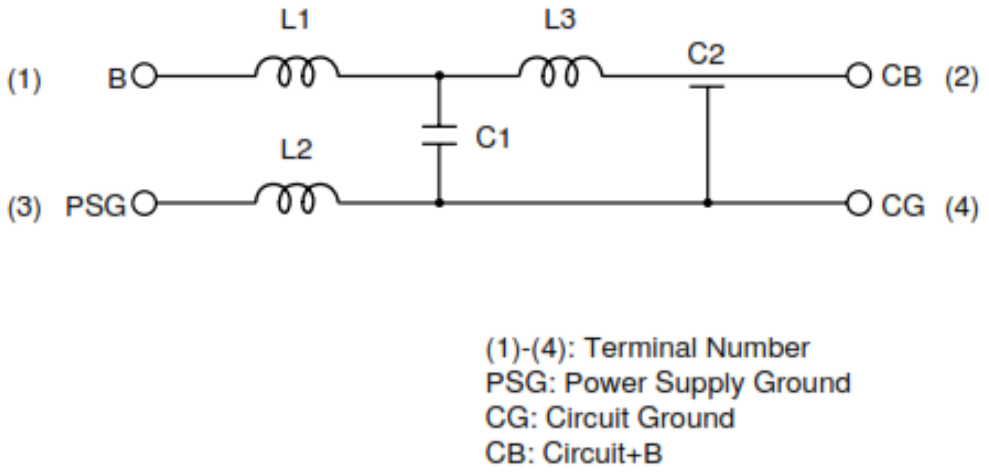
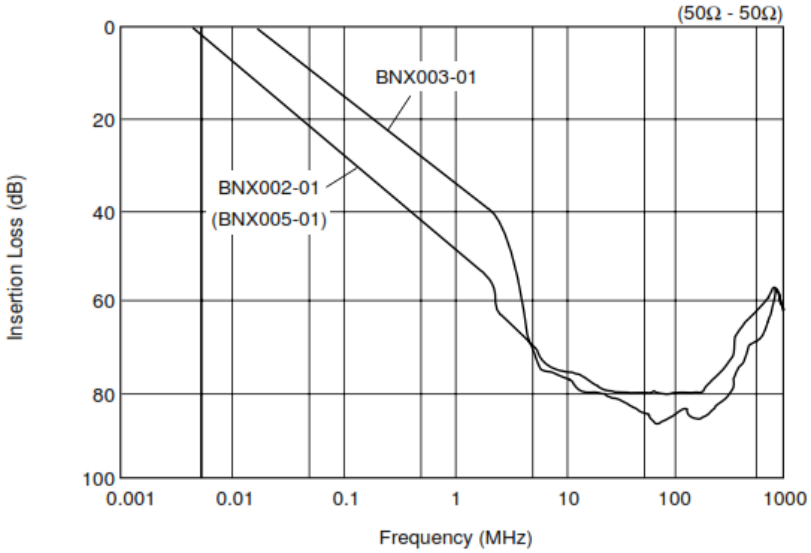


Figura 36: Esquema BNX002

Con este filtro se pueden eliminar ruidos procedentes de la red y de la fuente de alimentación. En concreto la fuente de alimentación produce un ruido y un rizado de un 1% de pico a pico en la banda de 20 MHz.



Gráfica 4: **Atenuación BNX002.**

Según esta gráfica el ruido producido por la fuente a 20 MHz tiene una atenuación de unos 75dB. También se incorporan dos condensadores electrolíticos de 4700µF y 2200µF a la entrada y a la salida del filtro para controlar los picos de tensión. A la salida obtenemos una señal de 5v filtrada.

El circuito típico que propone el fabricante para el conversor DC/DC MAX735 aparece en la figura 37.

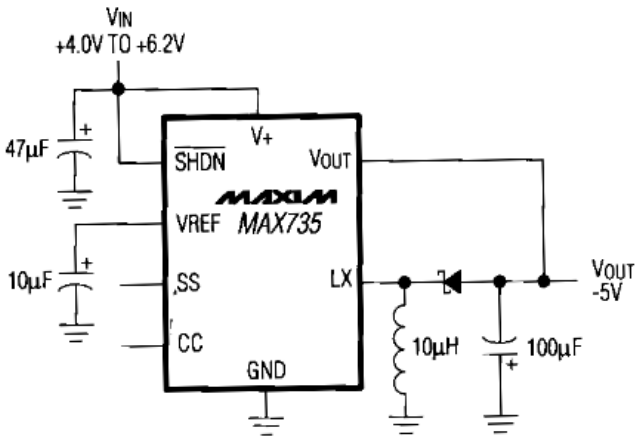


Figura 37: Circuito MAX735

La salida del convertidor se hace pasar por la misma etapa que en el circuito anterior, el filtro choke con los condensadores electrolíticos. Esto se hace para eliminar posibles ruidos de conmutación producidos por el conversor. Estas dos señales, +5v y -5v, serán las fuentes de alimentación para los distintos integrados del circuito.

5.8 Diseño PCB:

En este apartado se pueden visualizar la parte superior e inferior de la placa base diseñada con el programa de designSpark.

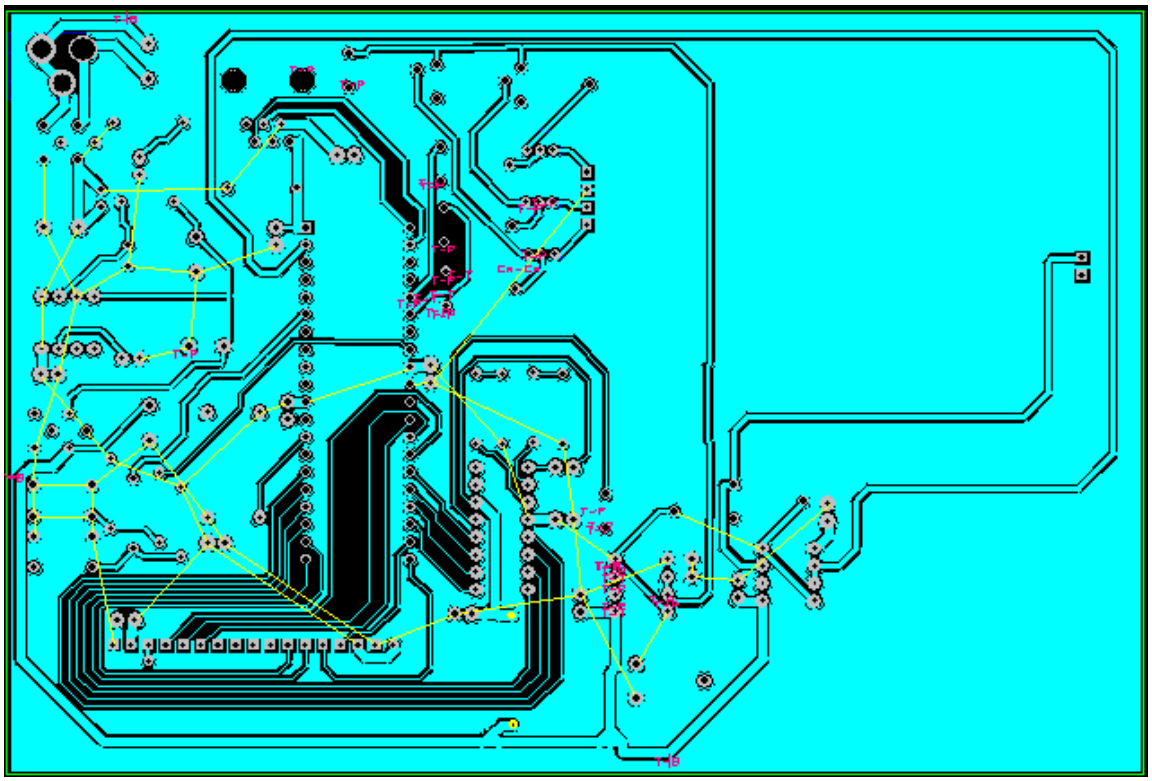


Figura 38: Parte inferior de la PCB diseñada

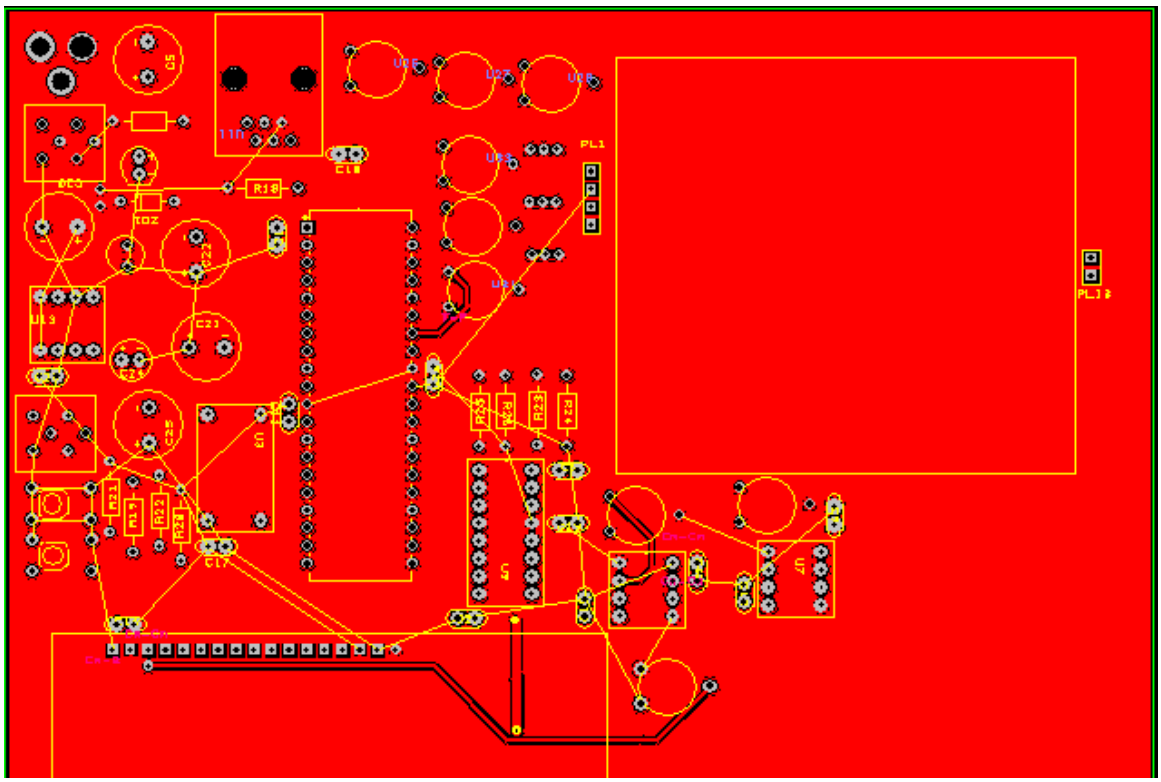


Figura 39: Parte superior de la PCB diseñada

La placa PCB se ha distribuido de forma que el usuario tenga fácil acceso y una correcta visualización de la pantalla LCD, de los pulsadores y del gran bloque de la parte superior que corresponde al porta cubetas estando todo en la parte superior de la placa. Las pistas se han intentado hacer por la parte inferior (color azul), ya que todos los componentes van soldados por esta parte. La alimentación entre 5v y masa se hace mediante capas, es decir toda la parte superior está a potencial 5v y la parte inferior está conectada a masa. La alimentación de -5v se ha hecho mediante una pista de más grosor con respecto a las de señal.

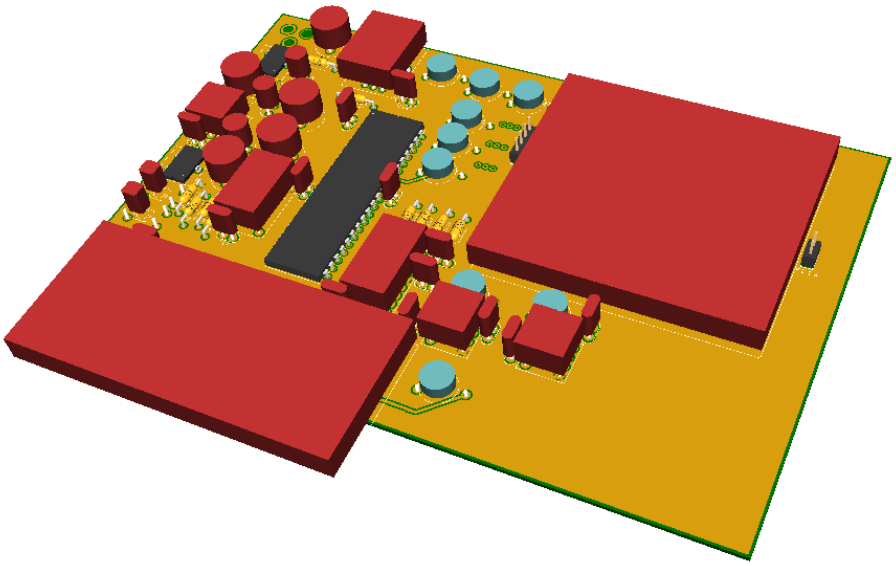


Figura 40: vista 3D de la PCB diseñada

Mejora del software del prototipo:

En este apartado se van a describir las modificaciones que ha sufrido el software del prototipo inicial para una correcta implementación de los distintos menús y la red neuronal.

5.9 Controlar RGB

Para el control del encendido y apagado de los leds se ha creado una función denominada encenderleds en la cual la variable led me va a marcar el encendido de los tres led en función de si su valor es 2,4 y 8 respectivamente. También se llama a otra función denominada encender1 en la que se pasa la variable led. En esta función se cargan los valores de potencia configurados para los led rojo, verde y azul a través del puerto C, se da la orden de encendido de los leds a través del puerto B y se activa el conversor analógico digital el cual recoge las medidas en la variable lectura de la blanca y posteriormente de la muestra.

5.10 Mejorar la configuración de la potencia de los LED

Para conseguir un mejor ajuste de la potencia de encendido de los leds se van a tener en cuenta las tensiones de saturación de los leds rojo, azul y verde siendo la tensión mínima a la que se van a alimentar que supondrá el 10% de la potencia Total y el 100% será de 5 voltios como se puede observar en la tabla 6 y también la variación entre estas referencias.

	Red(v)	Blue(v)	Green(v)
Vmax/100%	5	5	5
Vmin/10%	2,15	3,5	3,5
Δv/90%	2,85	1,5	1,5

Tabla 6

Por lo tanto si se varían las potencias desde el 10% hasta el 100% van a dar lugar a distintas tensiones en los leds como se ve en la tabla 7.

10%	2,15	3,50	3,50
20%	2,46	3,66	3,66
30%	2,78	3,83	3,83
40%	3,10	4,00	4,00
50%	3,41	4,16	4,16
60%	3,73	4,33	4,33
70%	4,05	4,5	4,50
80%	4,36	4,66	4,66
90%	4,68	4,83	4,83
100%	5,00	5,00	5,00

Tabla 7

Estas tensiones son traducidas a bits a través de la resolución que ofrece el conversor según la ecuación 18. En la tabla 8 se muestran los respectivos bits en decimal.

10% bits	**110**	**179**	**179**
20% bits	**126**	**188**	**188**
30% bits	**143**	**197**	**197**
40% bits	**159**	**205**	**205**
50% bits	**175**	**214**	**214**
60% bits	**191**	**222**	**222**
70% bits	**208**	**231**	**231**
80% bits	**224**	**239**	**239**
90% bits	**240**	**248**	**248**
100%bits	**255**	**255**	**255**

Tabla 8

Finalmente se traducen los valores de decimal a binario, que serán introducidos a los arrays denominados potenciaLedred y potenciaLedbluegreen

que se recorrerán con la variable pot [x3], la cual nos marca el nivel de potencia configurado para los distintos leds según x3. En la tabla 9 se ve tal conversión.

	Datos a Introducir en arrays		
10% bits	**1101110**	**10110011**	**10110011**
20% bits	**1111110**	**10111100**	**10111100**
30% bits	**10001110**	**11000100**	**11000100**
40% bits	**10011110**	**11001101**	**11001101**
50% bits	**10101111**	**11010101**	**11010101**
60% bits	**10111111**	**11011110**	**11011110**
70% bits	**11001111**	**11100110**	**11100110**
80% bits	**11011111**	**11101111**	**11101111**
90% bits	**11110000**	**11110111**	**11110111**
100%bits	**11111111**	**11111111**	**11111111**

Tabla 9

5.11 Simplificación del Menú de medida y Red

Partiendo del código de inicio únicamente se va a mantener similar la parte de medición de la absorbancia, ya que los otros parámetros de medición no se van a utilizar, por lo que van a ser eliminados.

Este código se va a depurar para poder integrar la función de red neuronal general y conseguir satisfacer los requisitos de memoria del punto de trabajo más crítico de la red.

El código presenta el diagrama de flujo de la figura 41.

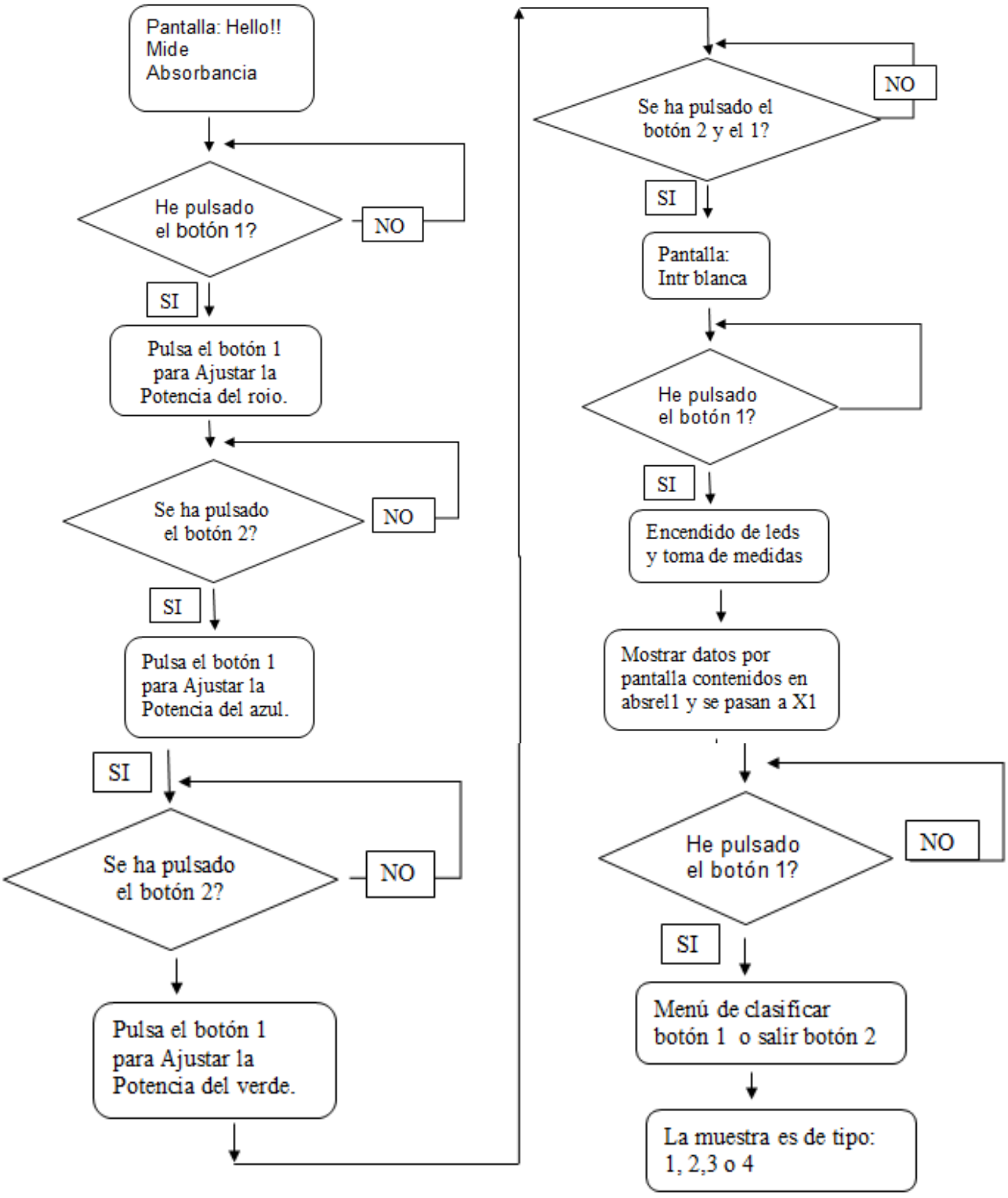


Figura 41: Diagrama de bloques del menú.

5.12 Resultados Finales:

Finalmente se ha tomado la decisión de que la red trabaje con 4 neuronas de salida y 3 neuronas en la capa oculta.

La red será capaz de distinguir las muestras según el colorante y si su concentración del 25% o 75% y así decir si es mayor o menor del 50%.

El resultado de la red nos indica el tipo de muestra medida, que se traduce en su color y concentración según la tabla 10.

Tipo muestra	0	1	2	3
Color-concentración	Roja 25%	Roja 75%	Azul 25%	Azul 75%

Tabla 10

Para llevar a cabo el entrenamiento de la red se realiza una base de datos con 10 mediciones por cada muestra tipo y posteriormente se observa el error de validación que se aprecia en la figura 42 para saber el grado de clasificación.

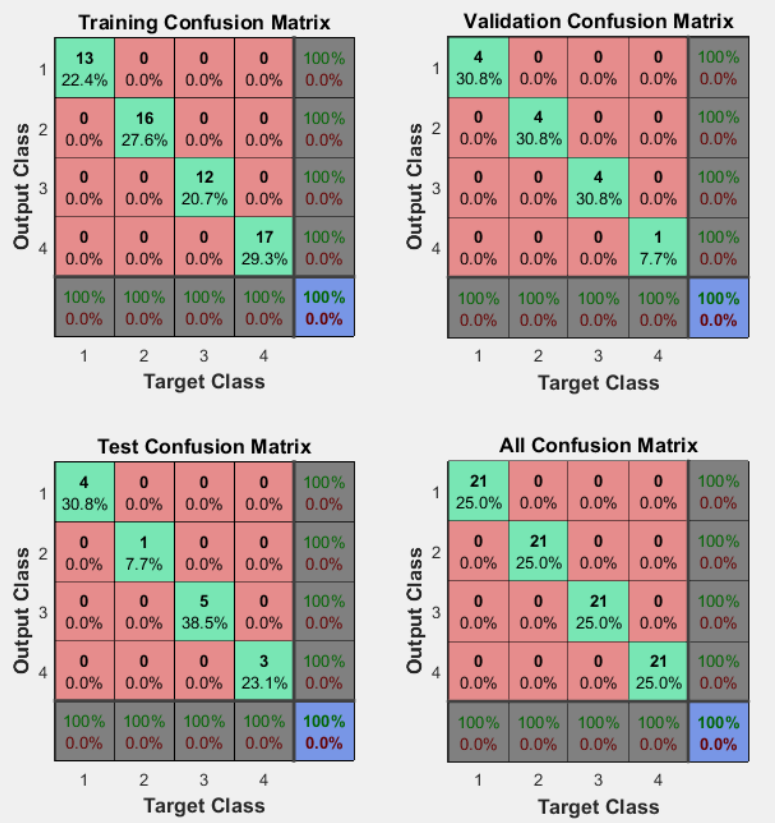


Figura 42

Como se puede apreciar, la red ha clasificado todas las muestras de manera satisfactoria. El siguiente paso es introducir las contantes en el código C.

Capítulo 6: Costes de los materiales

En la tabla 11 se encuentra la lista de materiales que han sido necesarios comprar para la realización del proyecto. El resto de material utilizados han sido adquiridos del laboratorio de electrónica básica de la UPNA y también mucho elementos han sido reutilizados del prototipo inicial.

REFERENCIA FABRICANTE	DESCRIPCION	PRECIO UNIDAD	Nº UNIDADES	TOTAL
L-154A4SURKQBDZGC	LED TRICOLOR	1,09	1	1,09 €
PIN-6DI	FOTODIODO	41,92 €	2	83,84 €
RTF-5010	MOLDURA LED	0,23 €	1	0,23 €
MHRJJ66NFRA	CONECTOR RJ11	1,33 €	1	1,33€
DAC0808LCN	CONVERSOR DIGITAL-ANALOGICO	1,98 €	1	1,98 €
FSM4JSMATR	PULSADOR	0,326 €	2	0,652 €
04M000000L638	OSCILADOR	6,80 €	1	6,80 €
2227MC-40-06-05-F1	ZÓCALO 40 PIN	1,38 €	1	1,38 €
2227MC-08-03-18-F1	ZÓCALO 16 PIN	0,734 €	1	0,734 €
808-AG11D-ESL-LF	ZÓCALO 8 PIN	0,699 €	3	2,097 €
MAX735EPA+	CONVERSOR DC/DC 5V→-5V	7,08 €	1	7,08 €
932-MIKROE-55	PANTALLA LCD	9,69 €	1	9,69 €
PIC16F877A-I/P	MICROCONTROLADOR	6,60 €	1	6,60 €
BNX002-01	FILTRO	8,86 €	2	17,72 €
MCEXT5V15WC1	FUENTE ALIMENTACION	16,98 €	1	16,98 €
MJ-179PH	CONECTOR ALIMENTACION	0,699 €	1	0,699 €
MCP6292-E/P	A.O.	0,91 €	3	2,73 €
RLB0712-100KL	BOBINA	0,261 €	1	0,261 €
711-1006	CONDENSADOR ELECTROLÍTICO 2200µF	0,268 €	2	0,536 €
ECA1AHG472	CONDENSADOR ELECTROLITICO 4700 µF	0,926 €	2	1,852 €
TOTAL				**163,76 €**

Tabla 11

Capítulo 7: Conclusiones y Líneas Futuras

En vista de los resultados obtenidos se han alcanzado las metas propuesta inicialmente en el proyecto.

Gracias a este trabajo se ha podido comprobar de forma práctica el uso de las redes neuronales artificales y su potencial de clasificación con medidas basadas en absorbancia óptica de manera satisfactoria.

Un elemento crítico a lo largo de todo el proceso ha sido la capacidad de memoria de datos del PIC. Gracias a la programación basada en punteros, se ha podido emplear la memoria de datos de manera eficiente. La red neuronal codificada en C es genérica: esto es, se puede dimensionar según las necesidades del usuario para tener diferentes números de neuronas en la capa oculta e identificar distintos tipos de muestras.

Además, la red aporta gran versatilidad, ya que, en función del entrenamiento que se le realice, es capaz de identificar distintos tipos de muestra; de esta forma, en el caso de tener que trabajar con muestras de otras características, basta con entrenar de nuevo la red para poder seguir utilizando el mismo hardware.

También este proyecto supone una mejora de la interfaz máquina-humano dando lugar a una mejor comprensión de los resultados obtenidos por el sistema de medida. En el caso del sistema obtenido, además de registrar las medidas de absorción, se especifica qué tipo de muestra se ha estudiado.

Como líneas futuras a este proyecto se pueden hacer mediciones con diferentes colores utilizando el mismo LED tricolor variando la programación del PIC y manteniendo el mismo hardware. Por otro lado, la información obtenida podría transmitirse por comunicación serie (puerto I_2C) o con la incorporación de comunicaciones inalámbricas como bluetooth, en cuyo caso podrían enviarse los datos obtenidos a un dispositivo móvil para su visualización.

Finalmente se abre la puerta para la programación de algoritmos más complejos de aprendizaje que reduzcan el margen de error y así permitan una mejor clasificación de las muestras.

Capítulo 8: Bibliografía

Diseño y desarrollo de un sistema optoeléctrico de medición a varias longitudes de onda (22/06/2015).
http://academicae.unavarra.es/xmlui/bitstream/handle/2454/19211/Memoria%20TFG%20Miguel%20Lara%20Arbizu.pdf?sequence=1&isAllowed=y

Desarrollo de una interfaz gráfica de redes neuronales usando Matlab (16/01/2016).
http://earchivo.uc3m.es/bitstream/handle/10016/8488/Proyecto%20Redes%20neuronales%20GUI.pdf?sequence=1

Matlab Help nprtool (5/11/2015).

Introducción a las redes neuronales para no expertos (10/25/2015) https://www.youtube.com/watch?v=uEIpuyUNvuA

Datasheet de components. (20/2/2016).
http://es.rs-online.com/web/ca/resumencesta/

Programación del PIC16f877A en C: Introducción al Hi-Tech Compiler sesión 1 de prácticas.

Manual htc pic manual.

Ebook PIC programming with C.

Pamplona a 10 de Marzo de 2016

Adrián Vicente Gómara

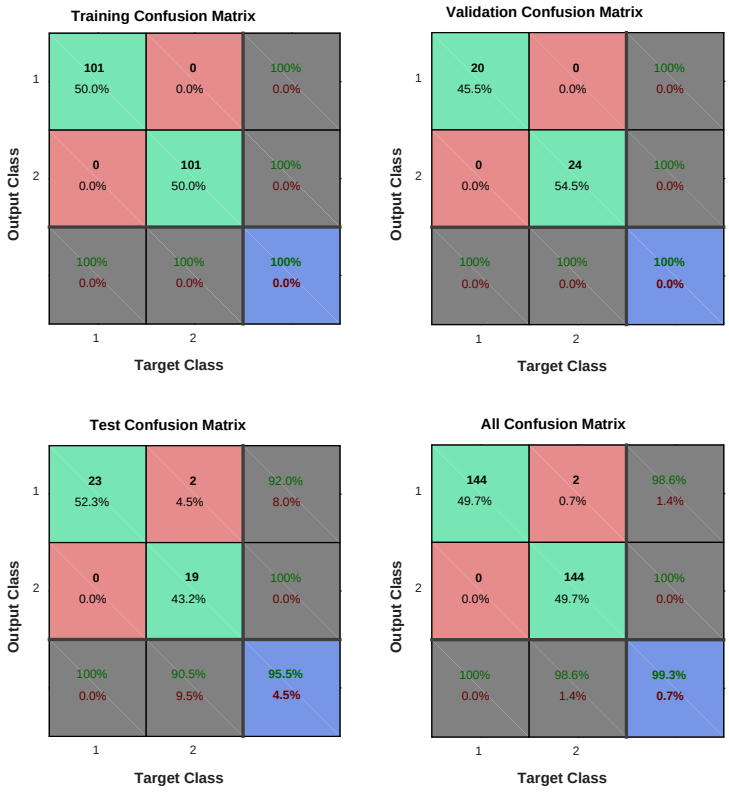
Anexo A

Estudio que determina el rendimiento de la red en función del número de neuronas de las distintas capas y el tamaño de la memoria de datos

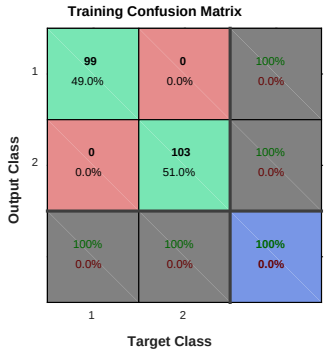
Para la realización de este estudio se parte de una base de datos que se encuentra en el apartado anexos, donde a partir de tres longitudes de onda se determina si la sustancia analizada es de color rojo o azul, si la concentración roja y azul es mayor del 50% o no y por último determinar si la concentración es del 25%, 50%, 75% y 100%. Para cada caso se ira variando el número de neuronas de la capa oculta y se observara el rendimiento de las clasificaciones. También se explicara el algoritmo de cálculo utilizado para determinar el tamaño utilizado en la memoria de datos de manera aproximada.

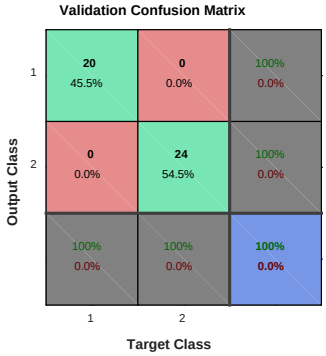
A.1 Caso1: Tres neuronas de entrada y dos neuronas de salida

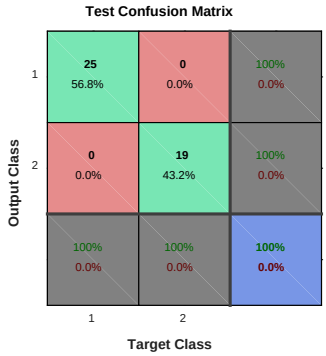
Entrada n	oculta X	salida m	Tamaño
3	1	2	32

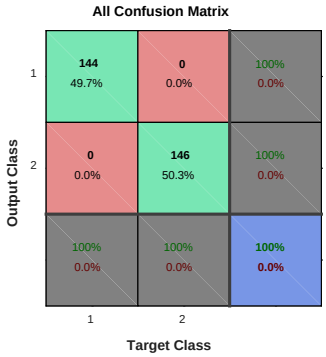


Entrada n	oculta X	salida m	Tamaño
3	2	2	56

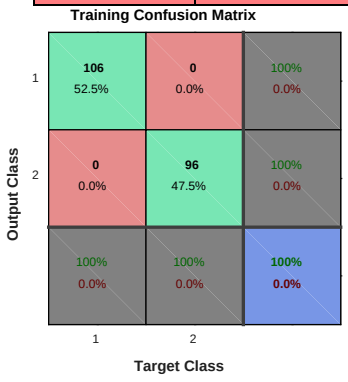


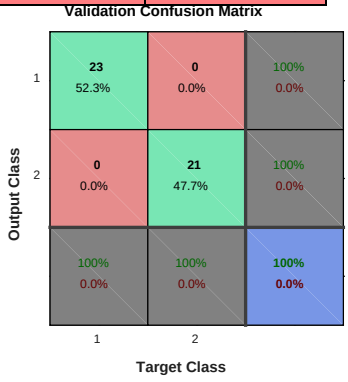


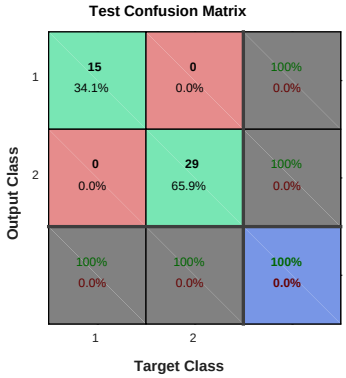


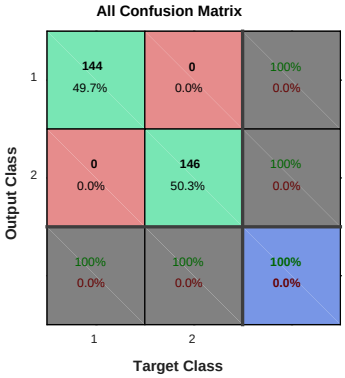


Entrada n	oculta X	salida m	Tamaño
3	3	2	80









Entrada n	oculta X	salida m	Tamaño
3	4	2	104

Entrada n	oculta X	salida m	Tamaño
3	5	2	128

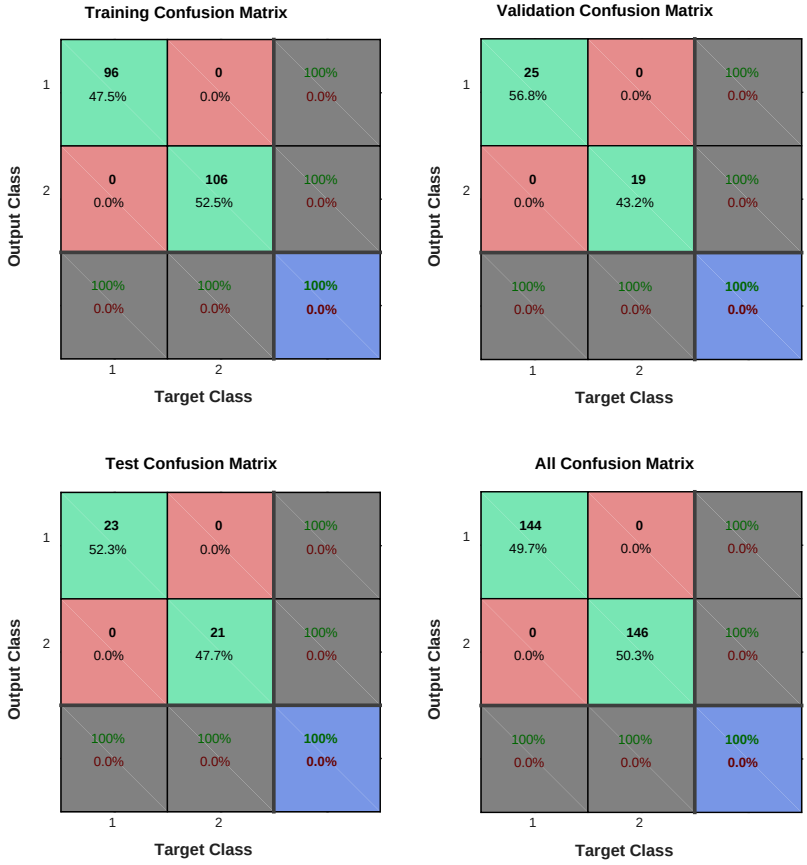
Entrada n	oculta X	salida m	Tamaño
3	6	2	152

Entrada n	oculta X	salida m	Tamaño
3	7	2	176

Entrada n	oculta X	salida m	Tamaño
3	8	2	200

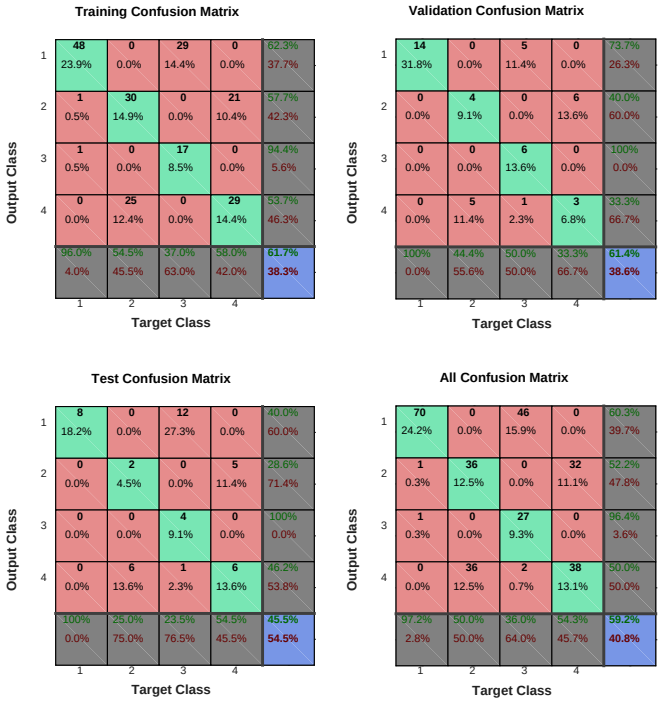
Entrada n	oculta X	salida m	Tamaño
3	9	2	224

Entrada n	oculta X	salida m	Tamaño
3	10	2	248

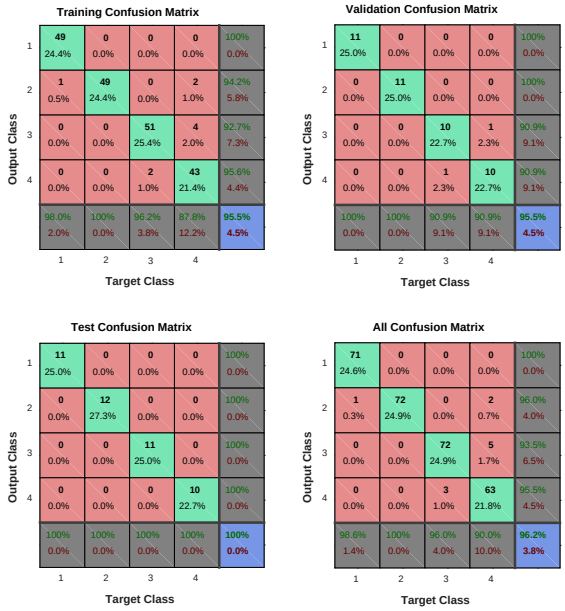


A.2 Caso 2: Tres neuronas de entrada y cuatro neuronas de salida.

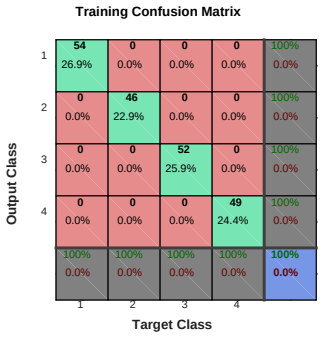
Entradas n	ocultas X	Salida m	Tamaño
3	1	4	48

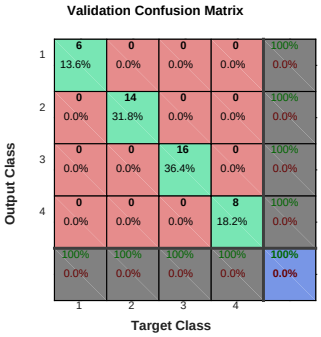


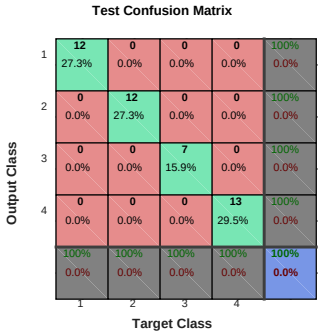
Entradas n	ocultas X	Salida m	Tamaño
3	2	4	80

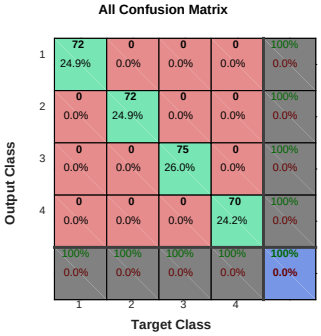


Entradas n	ocultas X	Salida m	Tamaño
3	3	4	112

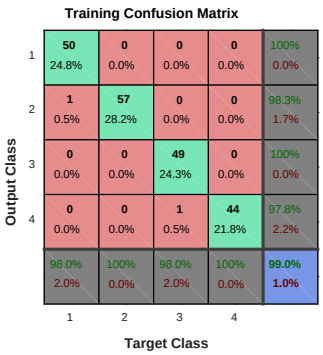


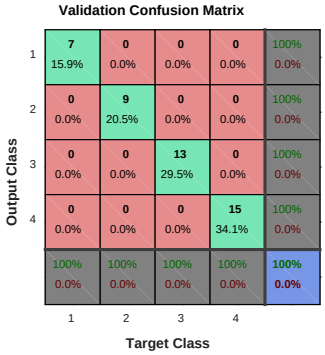


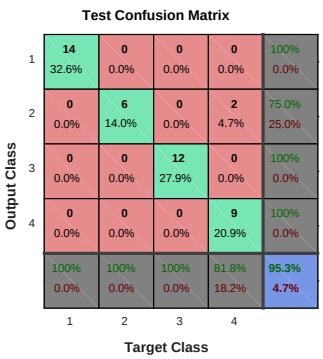


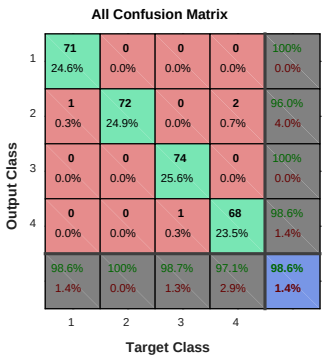


Entradas n	ocultas X	Salida m	Tamaño
3	4	4	144

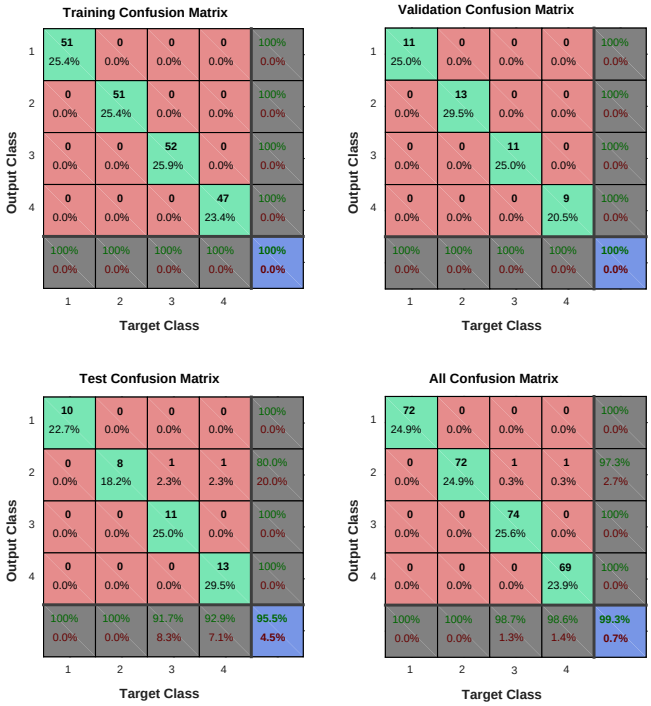




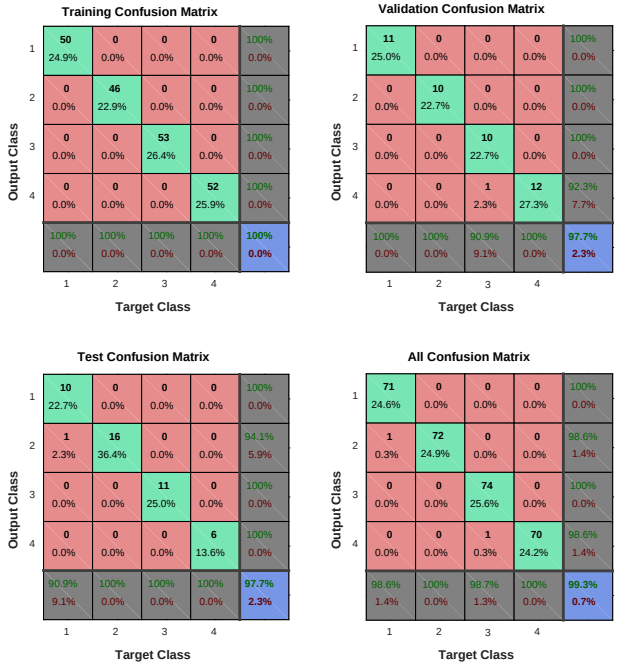




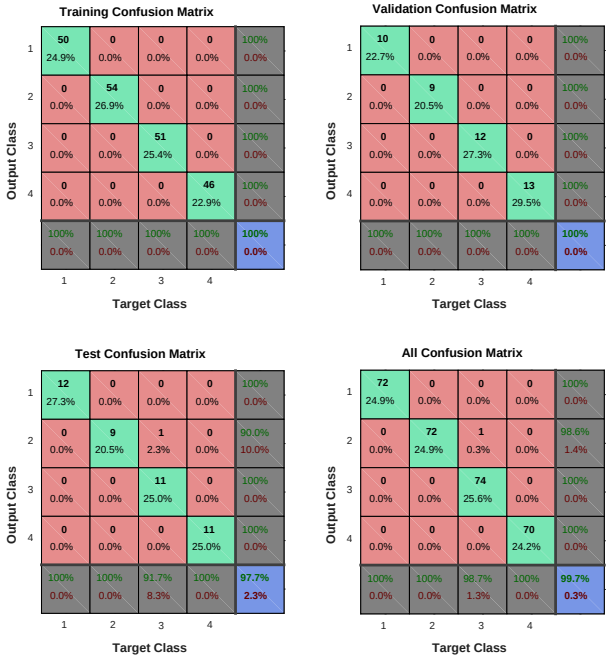
Entradas n	ocultas X	Salida m	Tamaño
3	5	4	176



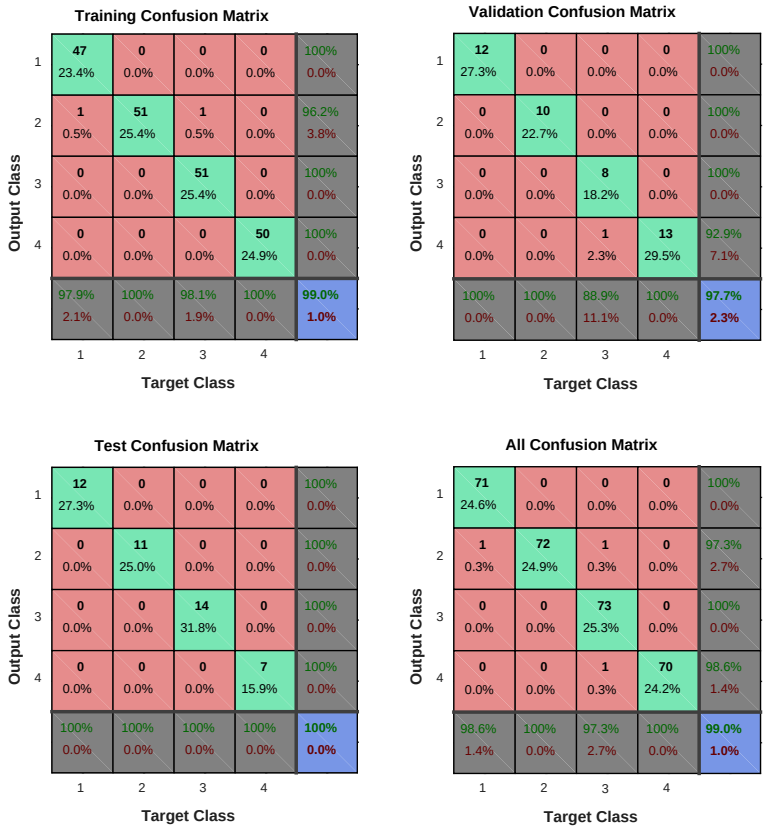
Entradas n	ocultas X	Salida m	Tamaño
3	6	4	208



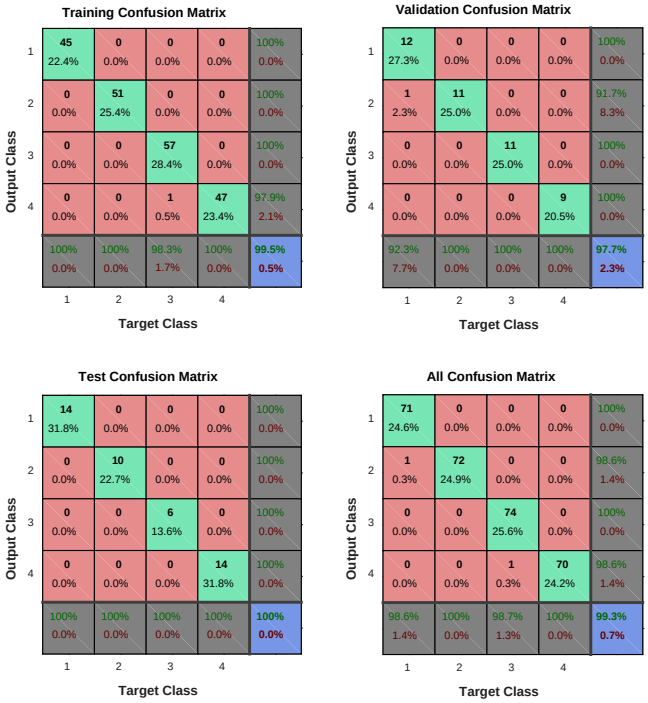
Entradas n	ocultas X	Salida m	Tamaño
3	7	4	240



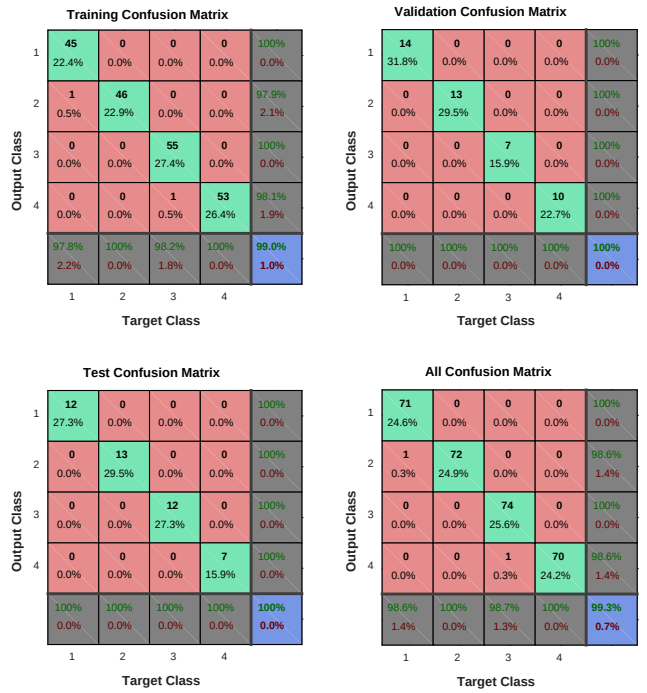
Entradas n	ocultas X	Salida m	Tamaño
3	8	4	272



Entradas n	ocultas X	Salida m	Tamaño
3	9	4	304

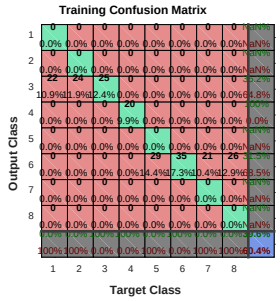


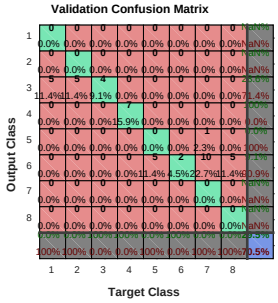
Entradas n	ocultas X	Salida m	Tamaño
3	10	4	336

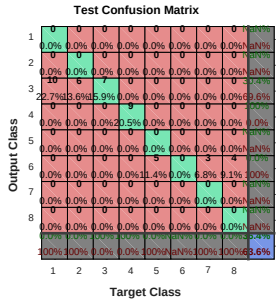


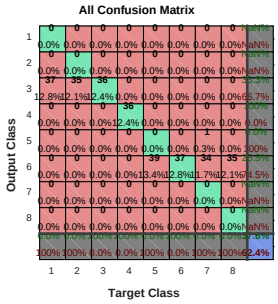
A.3 Caso 3: Tres neuronas de entrada y 8 neuronas de salida.

Entradas n	ocultas X	Salida m	Tamaño
3	1	8	80

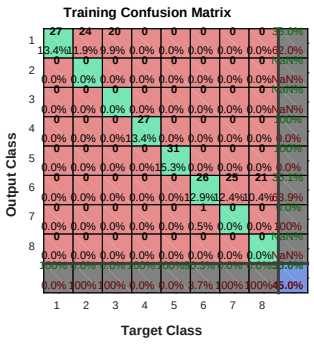


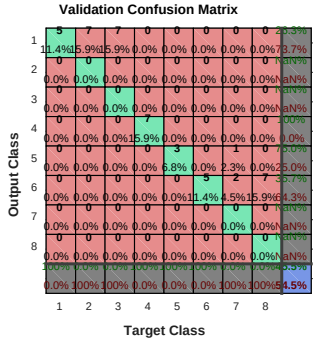


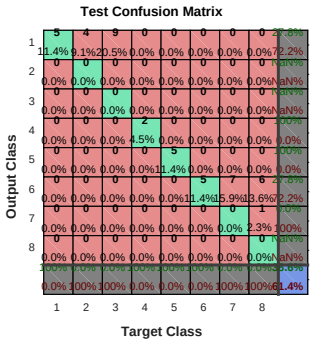


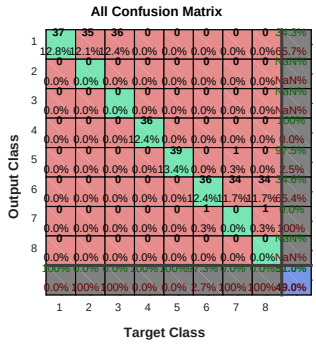


Entradas n	ocultas X	Salida m	Tamaño
3	2	8	128

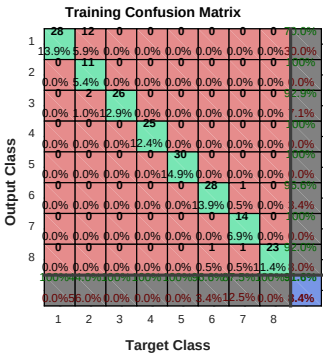


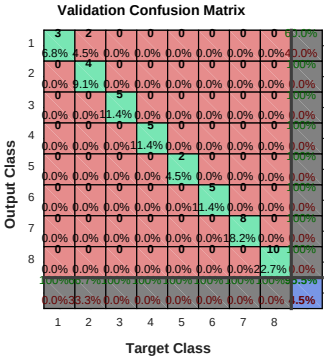


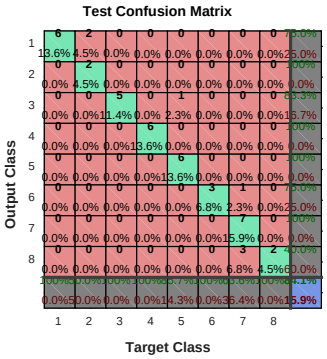


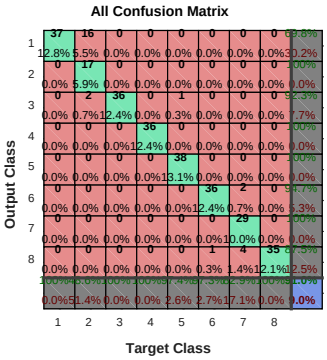


Entradas n	ocultas X	Salida m	Tamaño
3	3	8	176

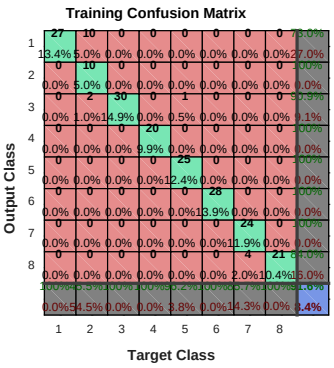


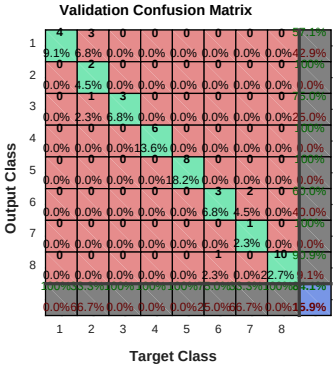


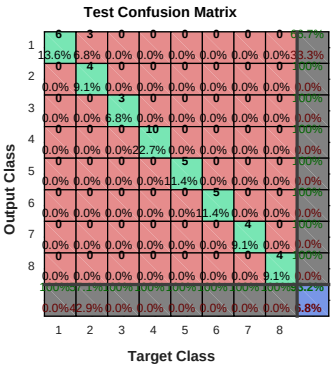


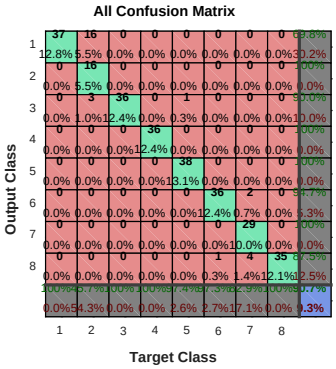


Entradas n	ocultas X	Salida m	Tamaño
3	4	8	224

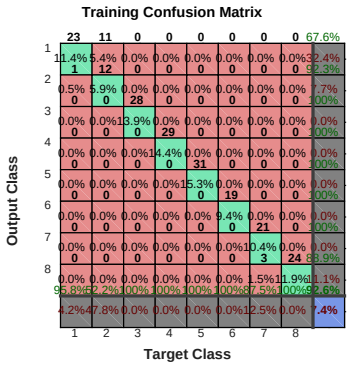


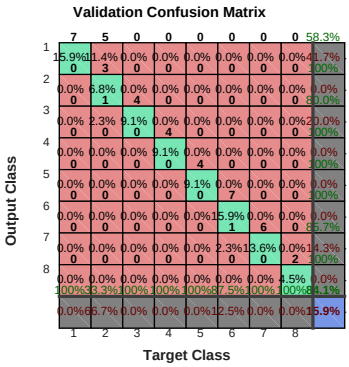


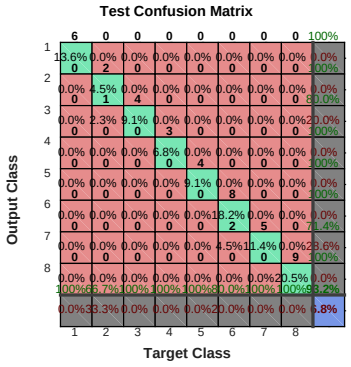


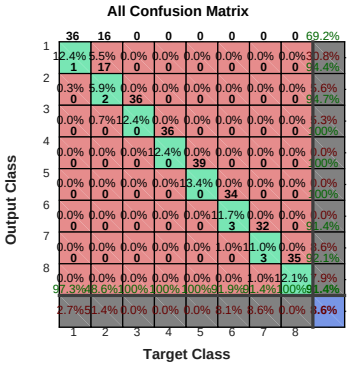


Entradas n	ocultas X	Salida m	Tamaño
3	5	8	272

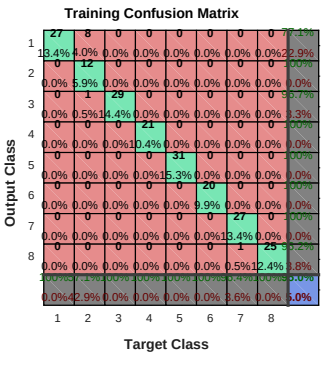


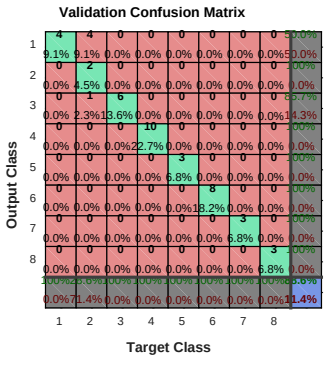


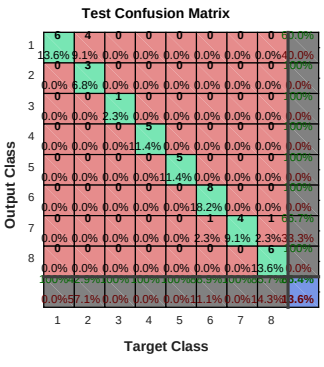


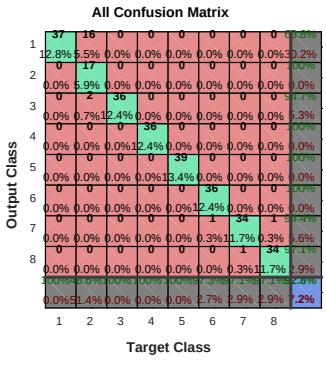


Entradas n	ocultas X	Salida m	Tamaño
3	6	8	320

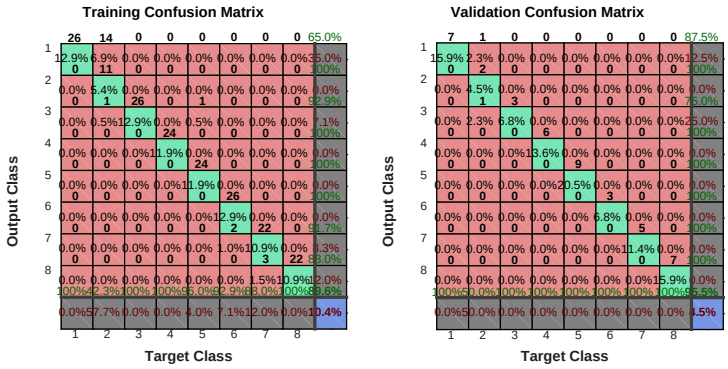


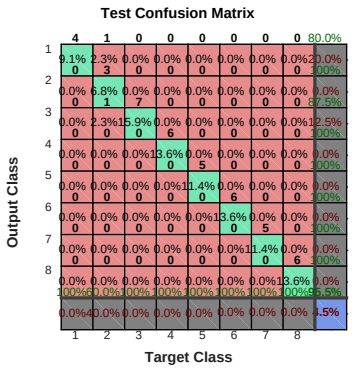


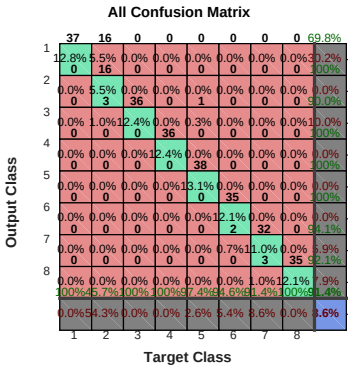




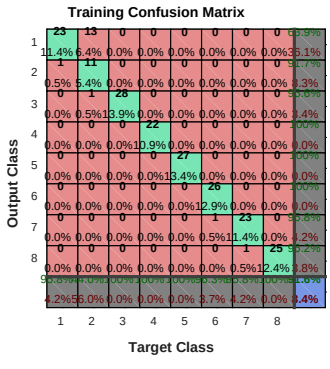
Entradas n	ocultas X	Salida m	Tamaño
3	7	8	368

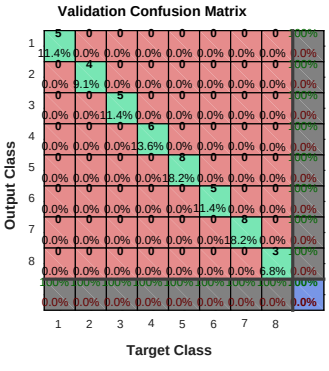


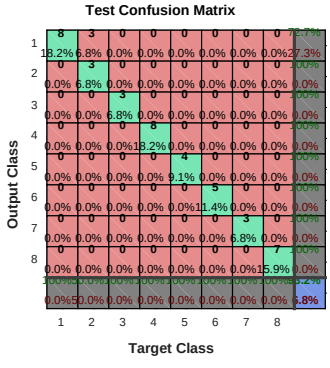


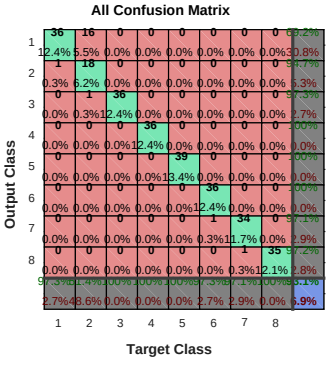


Entradas n	ocultas X	Salida m	Tamaño
3	8	8	416

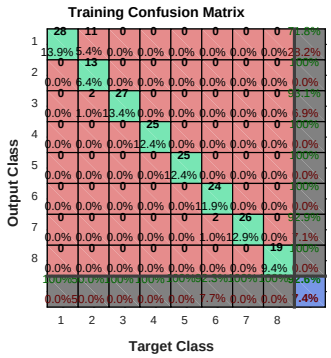


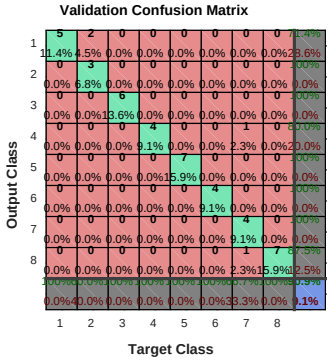


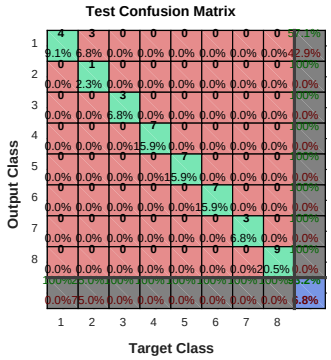


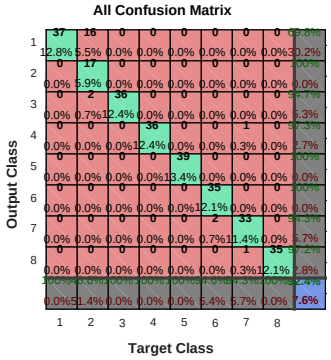


Entradas n	ocultas X	Salida m	Tamaño
3	9	8	464

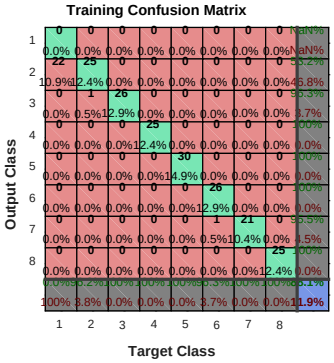


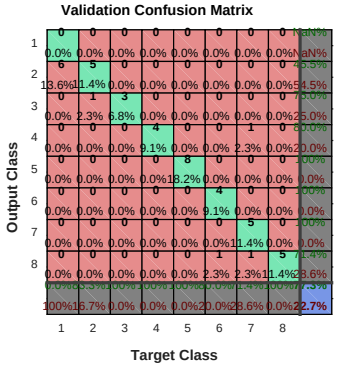


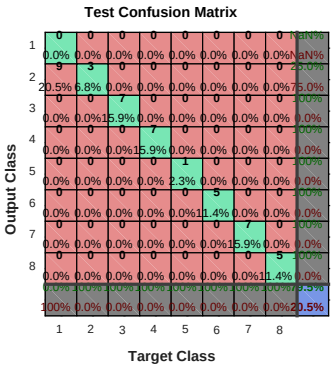


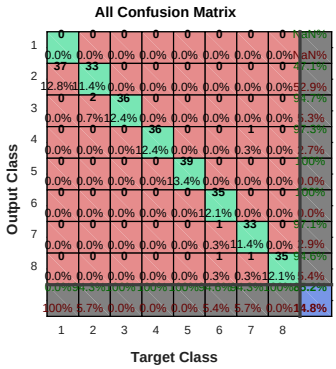


Entradas n	ocultas X	Salida m	Tamaño
3	10	8	512









A.4 Análisis de los resultados obtenidos:

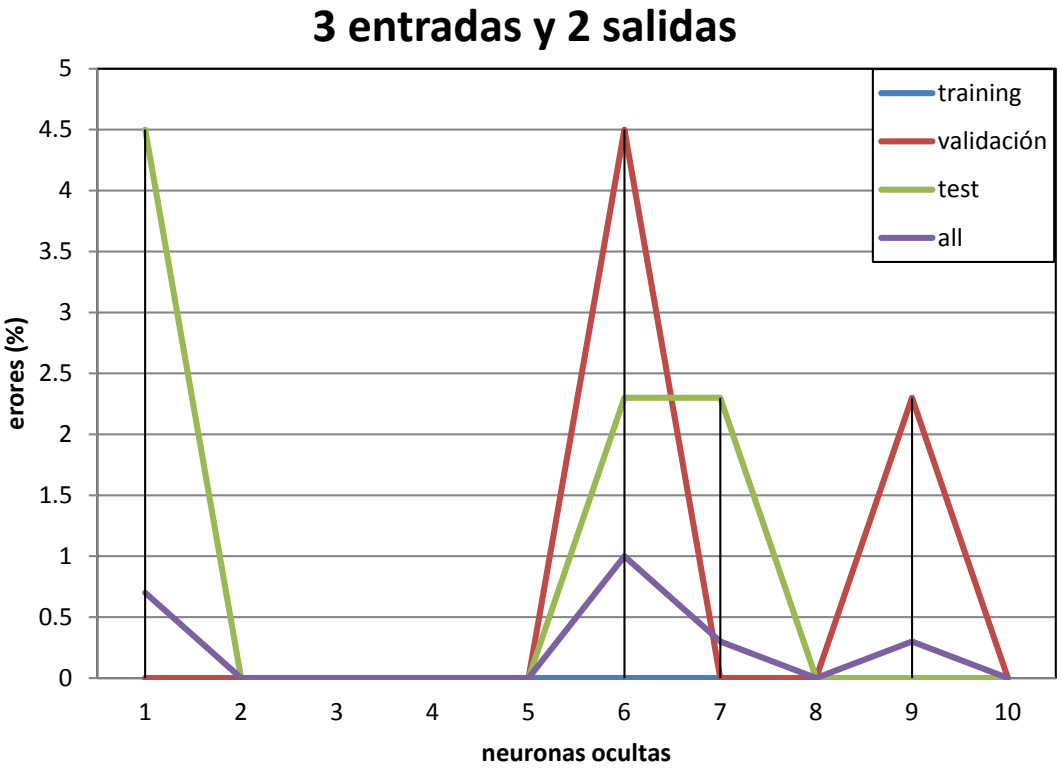
En este apartado se analizaran los errores de entrenamiento, validación, test y all que se diferencian por el número de muestras utilizadas para determinar su respectivo error. En la tabla 11 se observan todas las combinaciones estudiadas y los puntos críticos que se dan en función del número de neuronas ocultas en cada sector de neuronas de salida.

#Neuronas			Errores				
Entrada	Salida	Ocultas	Training (%)	Validatión (%)	Test (%)	All (%)	Memoria
3	2	1	0	0	4,5	0,7	32
3	2	2	0	0	0	0	56
3	2	3	0	0	0	0	80
3	2	4	0	0	0	0	104
3	2	5	0	0	0	0	128
3	2	6	0	4,5	2,3	1	152
3	2	7	0	0	2,3	0,3	176
3	2	8	0	0	0	0	200
3	2	9	0	2,3	0	0,3	224
3	2	10	0	0	0	0	248
3	4	1	38,3	38,6	54,5	40,8	48
3	4	2	4,5	4,5	0	3,8	80
3	4	3	0	0	0	0	112
3	4	4	1	0	4,7	1,4	144
3	4	5	0	0	4,5	0,7	176
3	4	6	0	2,3	2,3	0,7	208
3	4	7	0	0	2,3	0,3	240
3	4	8	1	2,3	0	1	272
3	4	9	0,5	2,3	0	0,7	304
3	4	10	1	0	0	0,7	336
3	8	1	60,4	70,5	63,6	62,4	80
3	8	2	45	54,5	61,4	49	128
3	8	3	8,4	4,5	15,9	9	176
3	8	4	8,4	4,5	6,8	9,3	224
3	8	5	7,4	15,9	6,8	8,6	272
3	8	6	5	11,41	13,6	7,2	320
3	8	7	10,4	4,5	4,5	8,6	368
3	8	8	8,4	0	6.8	6,9	416
3	8	9	7,4	9,1	6,8	7,6	464
3	8	10	11,9	22,7	20,5	14,8	512

Tabla 12

A.4.1 Caso1: Tres neuronas de entrada y dos neuronas de salida.

En la gráfica 1 se puede ver como el error de validación alcanza su máximo para 6 neuronas ocultas y se dispara para 9 neuronas. Para el error de test alcanza su máximo en 1 neurona oculta y llega hasta 2,3 % para 6 y 7 neuronas. En el error total se dan pequeños picos para 1, 6,7 y 9 neuronas. Por el otro lado se observa un error nulo en el rango de 2 a 5 neuronas considerándose como una zona adecuada para trabajar.



Gráfica 5

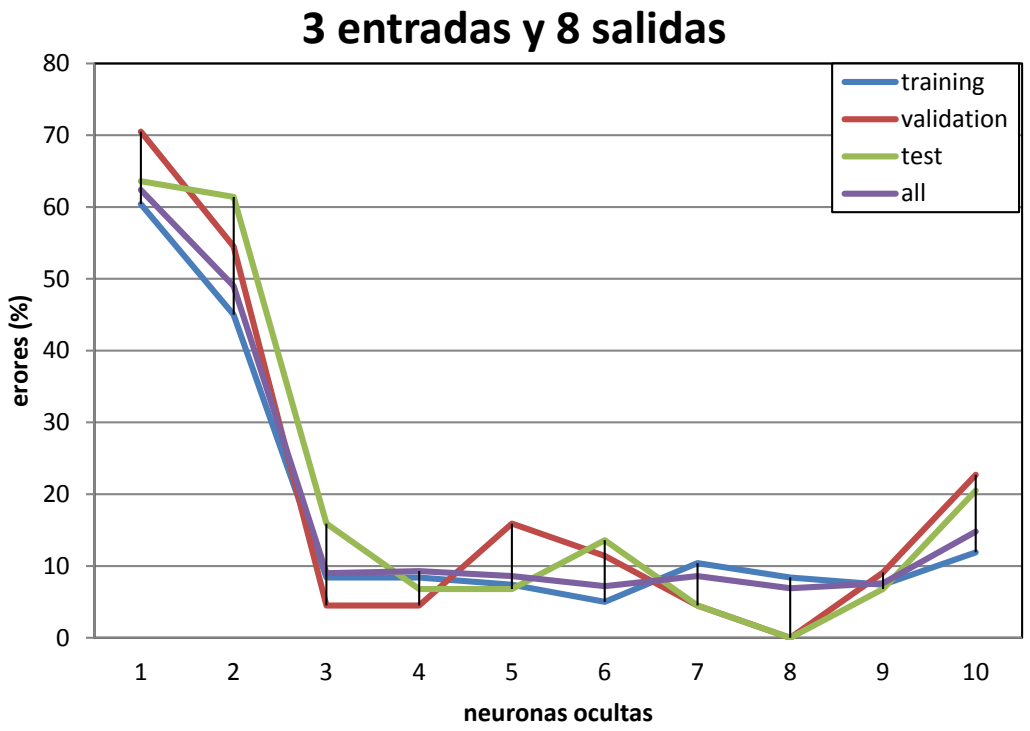
A.4.2 Caso 2: Tres neuronas de entrada y cuatro neuronas de salida.

En la gráfica 2 se observa cómo se disparan todos los errores para el caso de 1 neurona y luego se incrementan de manera muy débil en el rango de 4 a 9. En este caso se trabajara con 3 y 10 neuronas ocultas.

Gráfica 6

A.4.3 Caso 3: Tres neuronas de entrada y 8 neuronas de salida.

En la gráfica 3 se observa un incremento de todos los errores para 1 neurona llegando hasta el 70% en el error de validación, posteriormente estos van decreciendo hasta que alcanza la neurona 3 y a partir se estabilizan en una franja el 15% al 0% para las neuronas siguientes. El punto óptimo de trabajo se considerara para 4, 8 y 9 neuronas ocultas.



A.5 Algoritmo de Cálculo para determinar el tamaño de memoria

El cálculo de almacenamiento de la memoria de datos es aproximado y sigue el sigue la ecuación 7.

$$bytes = 4[(n_0 \cdot n_1 + n_1) + (n_2 \cdot n_1 + n_2)]$$

$siendo\ n_0: número\ de\ neuronas\ de\ entrada;\ \ n_1: número\ de\ neuronas\ ocultas\ y$

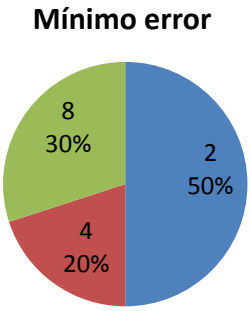
$$n_2: número\ de\ neuronas\ de\ salida$$

Ecuación 22

A.6 Conclusiones:

Tras el estudio realizado se puede concluir que:

- En el caso de 2 neuronas de salida se trabajara en el rango de 2 a 5 neuronas ocultas.
- En el caso de 4 neuronas de salida se trabajara con 3 y 10 neuronas ocultas.
- En el caso de 8 neuronas de salida se trabajara con 4, 8 y 9 neuronas ocultas.



Anexo B

Estudio del código que utiliza Matlab para generar las salidas de la red neuronal

B.1 Objetivo:

El objetivo de este informe se basa en entender el código que utiliza Matlab para la generación de las salidas de la red neuronal a partir de una muestra conocida y donde los pesos y bias son obtenidos del entrenamiento de la red con Matlab. Para el entrenamiento de la red neuronal se introduce una matriz de 290 muestras x 3 neuronas en los input, una matriz de 290 muestras x 2 neuronas en los output y 2 neuronas en la capa oculta, ya que en estudios previos se ha podido observar que en esta disposición el error de desviación es muy pequeño. Muestra seleccionada para entender el código:

λ1	λ2	λ3	Rojo	Azul
0.002	2.681	3	1	0

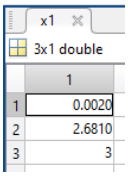
Nota: la coma [,] se designara como punto [.] ya que Matlab lo interpreta de esta forma.

B.2 Explicación Didáctica:

B.2.1 Paso1

Se introduce la matriz de la muestra a analizar: x1=[0.002 2.681 3]';

Nota: esta matriz tiene que ser traspuesta para ello se utiliza [… '…] la comilla para que aparezca como 3 filas y una columna.



B.2.2 Paso 2:

¿Cuántas columnas tiene la matriz x1?

Se sabe que Q=1 por lo que si se introduce Q = size(x1,2);

B.2.3 Paso 3:

Se entra en la función:

```
% Input 1
xp1 = mapminmax_apply(x1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);
```

Que hace una llamada a la subrutina que contiene la función **mapminmax_apply y entramos a esa rubrutina:**

```
% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
y = bsxfun(@minus,x,settings_xoffset);
y = bsxfun(@times,y,settings_gain);
y = bsxfun(@plus,y,settings_ymin);
end
```

Lo que ocurre es que renombra los valores de xp1=y; x=x1; settings_gain= x1_step1_gain y asi sucesivamente.

A continuación se analiza lo que ocurre en esa subrutina paso a paso para las variables Xp1,x1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin.

```
>> Xp1 = bsxfun(@minus,x1,x1_step1_xoffset);
```

Xp1 “mayúscula X”

Según la teoría este código realiza la siguiente ecuación:

$$Xp1 = x1 - x1\_step\_xoffset$$

Comprobación práctica:

x1

3x1 double

	1
1	0.0020
2	2.6810
3	3

x1 x1_step1_xoffset

3x1 double

	1	2
1	0	
2	0	
3	0.0520	

x1 x1_step1_xoffset Xp1

3x1 double

	1	2	3
1	0.0020		
2	2.6810		
3	2.9480		

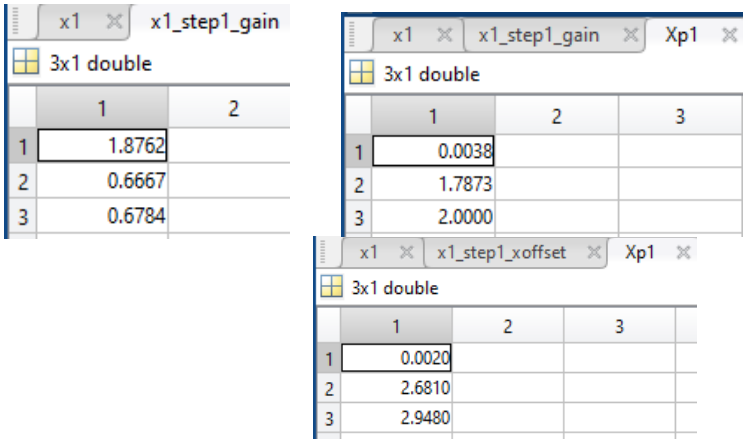
El resultado es correcto.

```
>> Xp1=bsxfun(@times,Xp1,x1_step1_gain);
```

Según la teoría este código hace la siguiente ecuación:

$$Xp1 = Xp1 * x1\_step\_gain$$

Comprobación práctica:



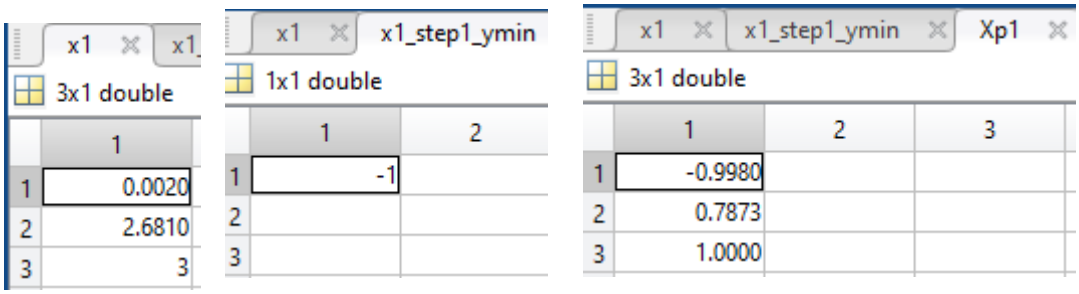
Las operaciones son correctas.

```
>> Xp1=bsxfun(@plus,Xp1,x1_step1_ymin);
```

Según la teoría este código efectúa la siguiente ecuación:

$$Xp1 = Xp1 + x1\_step\_ymin$$

Comprobación práctica:



Las operaciones que hace son correctas.

B.2.4 Paso 4:

Posteriormente tras analizar el código se entra a la siguiente función:

```
% Layer 1
a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);
```

Que hace una llamada a la subrutina que contiene la función **tansig_apply** y entramos a esa rubrutina:

```
% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
a = 2 ./ (1 + exp(-2*n)) - 1;
end
```

Lo que ocurre es que renombra los valores de n= repmat(b1,1,Q) + IW1_1*xp1 yo voy a llamar a la variable n como sumatori_capa1 y IW1_1*xp1= producto_capa1.

```
>> producto_capa1=IW1_1*xp1;
Undefined function or variable 'xp1'.

Did you mean:
>> producto_capa1=IW1_1*Xp1;
```

Como se puede ver he escrito mal Xp1 y Matlab me ha dicho si quería decir otra cosa. “gracias”.

Comprobación de esta operación de multiplicación:

IW1_1

2x3 double

	1	2	3
1	-2.2915	1.2104	-0.2602
2	-1.2055	1.1563	1.6327

Xp1

3x1 double

	1	2
1	-0.9980	
2	0.7873	
3	1.0000	

producto_capa1

2x1 double

	1	2	3
1	2.9797		
2	3.7462		

2x3 3x1 2x1

$$-2.2915 * -0.9980 + 1.2104 * 0.7873 + -0.2602 * 1 = 2.9797$$

La operación es correcta.

Posteriormente se llama a la otra parte de la función sumatorio_capa1 como via_capa1

```
>> via_capa1=repmat(b1,1,Q);
```

Nota: se puede cambiar el nombre de la variable desde el workspace.

Donde la función repmat lo que hace es repetir la matriz b1, en la dimensión de filas, Q veces.

Si se introduce:

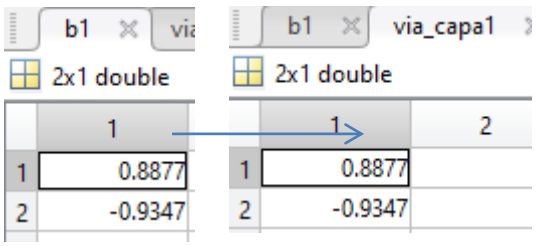
```
>> help repmat
repmat - Repeat copies of array

    This MATLAB function returns an array containing n copies of A in the row and
    column dimensions.
```

Traducción: esta función devuelve una matriz de n copias de A in la dimensión de filas y columnas.

```
B = repmat(A,n)
B = repmat(A,r1,...,rN)
B = repmat(A,r)
```

Donde r1 es la dimensión de filas o colunmas y rN son las veces a repetir en nuestro caso Q=1

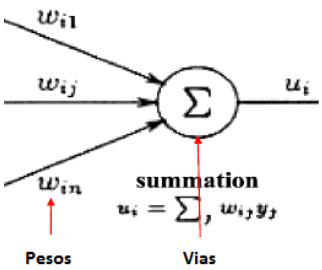
Comprobación:



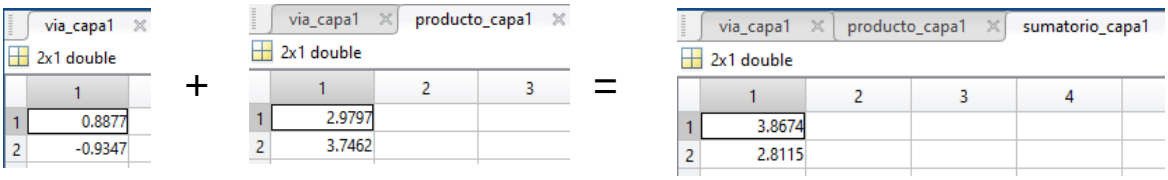
Se observa como el resultado es el mismo para este caso, pero si no lo fuera la matriz se repetiría en filas.

Seguidamente se crea una variable llamada sumatorio_capa1 para representar la suma de via_capa1 con producto_capa1.

```
>> sumatorio_capa1=(via_capa1+producto_capa1);
```



Comprobación:



Se puede observar que es correcto.

A continuación se analiza lo que ocurre en la subrutina de la función **tansig_apply** paso por paso para la nueva variable de sumatorio_capa1.

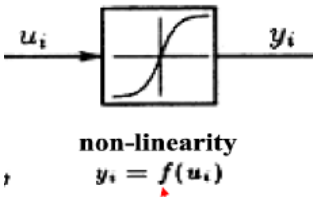
```
>> a1 = 2 ./ (1 + exp(-2*sumatorio_capa1)) - 1;
```

Se almacena la función en a1.

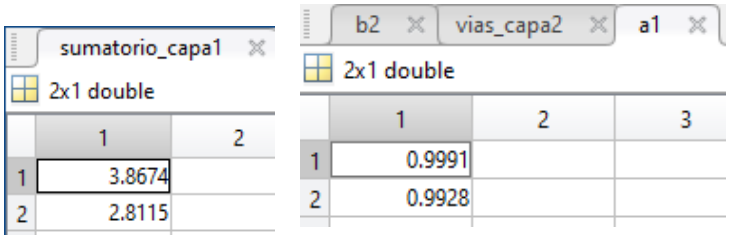
Este código viene a representar la siguiente operación:

$$a1 = \frac{2}{1 + e^{-2*sumatorio\_capa1}} - 1$$

Se Satura el valor de sumatorio_capa1 a unos valores concretos. Si el sumatorio de capa_1 tiende a infinito la exponencial tendera a cero y por lo tanto se saturara en 1.Si por el contrario sumatorio_capa1 tiende a menos infinito el cociente tendera a cero y por lo tanto se satura en -1. En la siguiente ilustración se muestra su efecto.



Comprobación:



Se aprecia como los valores de “a1” estan saturando hacia 1.

B.2.5 Paso 5:

A continuación se llega a la siguiente función:

```
% Layer 2
a2 = softmax_apply(repmat(b2,1,Q) + LW2_1*a1);
```

Que lleva a la subrutina de esa función:

```
% Competitive Soft Transfer Function
function a = softmax_apply(n)
nmax = max(n,[],1);
n = bsxfun(@minus,n,nmax);
numer = exp(n);
denom = sum(numer,1);
denom(denom == 0) = 1;
a = bsxfun(@rdivide,numer,denom);
end
```

Pero en este caso la n que se muestra va a ser repmat(b2,1,Q)+ LW2_1*a1 y se va a llamar sumatorio_capa2 y de la misma manera que anteriormente se hará lo mismo con las dos partes que la forman:

```
>> vias_capa2=repmat(b2,1,Q);
```

Comprobación:

b2

2x1 double

	1
1	0.8296
2	-0.1004

b2 vias_capa2

2x1 double

	1	2
1	0.8296	
2	-0.1004	

podemos ver como es correcta la matriz según el criterio explicado anteriormente de la función repmat

```
>> producto_capa2=LW2_1*a1;
```

a1 LW2_1

2x2 double

	1	2
1	1.4762	1.5700
2	-1.4711	-1.1591

X

a1

2x1 double

	1
1	0.9991
2	0.9928

=

a1 LW2_1 producto_capa2

2x1 double

	1	2	3
1	3.0336		
2	-2.6205		

2x2 2x1 2x1

$$1.4762 * 0.9991 + 1.5700 * 0.9928 = 3.0336$$

La operación es correcta.

```
>> sumatorio_capa2=vias_capa2+producto_capa2;
```

vias_capa2

2x1 double

	1
1	0.8296
2	-0.1004

vias_capa2 producto_capa2

2x1 double

	1	2	3
1	3.0336		
2	-2.6205		

vias_capa2 producto_capa2 sumatorio_capa2

2x1 double

	1	2	3	4	5
1	3.8632				
2	-2.7209				

Comprobación: Se ve que la suma es correcta.

A posteriori se analiza lo que ocurre en la subrutina de la función paso a paso para la nueva variable de sumatorio_capa2.

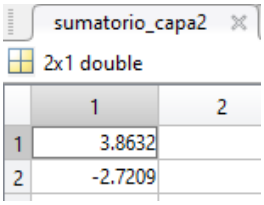
```
>> nmax = max(sumatorio_capa2,[],1);
```

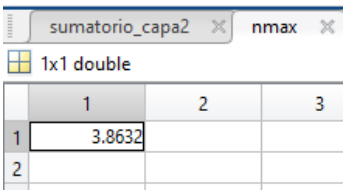
La función **max** dice que:

```
max - Largest elements in array

    This MATLAB function returns the largest elements of A.

    M = max(A)
    M = max(A,[],dim)
```

Traducción: da el número más grande de la matriz en la dimensión filas.

Comprobación:





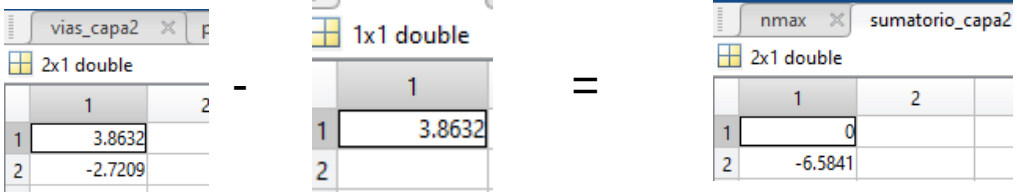
La dimensión filas se recorre siguiendo la flecha y el resultado es correcto.

```
>> sumatorio_capa2=bsxfun(@minus,sumatorio_capa2,nmax);
```

Esta función va a realizar la siguiente operación

Sumatorio_capa2=sumatorio_capa2-nmax

Comprobación:

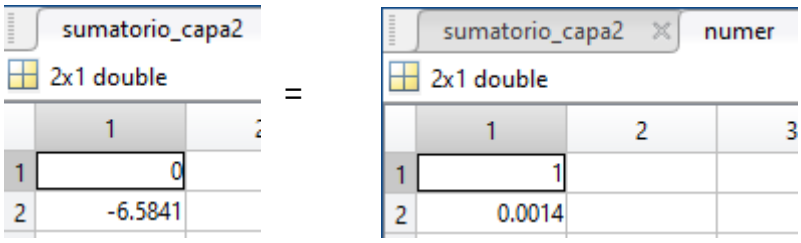


El resultado es correcto.

```
>> numer = exp(sumatorio_capa2);
```

Esta función va a realizar la siguiente operación: $numer = e^{sumatorio\ capa2}$

Comprobación:



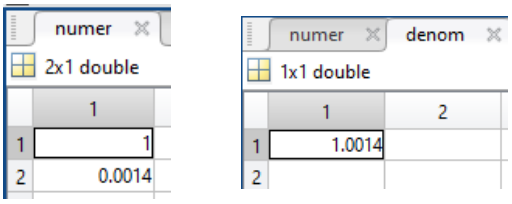
El resultado es correcto.

```
>> denom = sum(numer,1);
```

Esta función quiere decir que va a sumar los valores de la matriz numer en la dimensión de recorrido filas. Véase en:

```
sum - Sum of array elements

    This MATLAB function returns the sum of the elements of A along the first array
    dimension whose size does not equal 1.

    S = sum(A)
    S = sum(A,dim)
```

Comprobación:

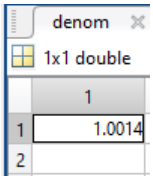


Como se puede ver, el resultado es correcto.

```
>> denom(denom == 0) = 1;
```

Este código es condicional. Si denom=0 entonces pon denom a 1. En nuestro caso no varía.

Comprobación:

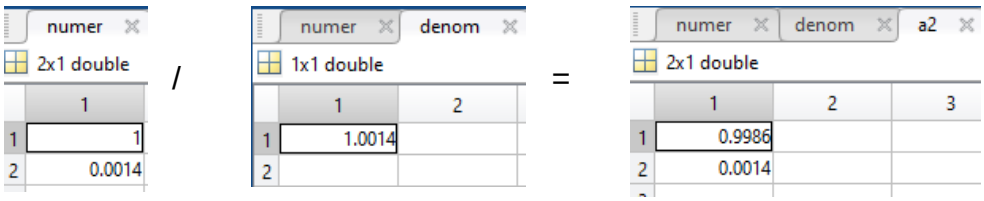


El resultado es correcto.

```
>> a2 = bsxfun(@rdivide,numer,denom);
```

Esta función va a realizar la siguiente operación: $a2 = \frac{numer}{denom}$

Comprobación:



El resultado obtenido es correcto.

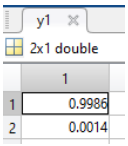
B.2.6 Paso 6:

Finalmente se designa:

```
% Output 1
y1 = a2;
end
```

```
>> y1 = a2;
```

salida es



Comprobación correcta, nuestra (1;0).

En el apartado de anexos se puede observar la generación de todo el código que aparece en la ventana de comandos de Matlab.

Anexo C

Explicación del algoritmo matemático programado en C para los distintos casos y las estructuras necesarias.

A.1 ANN para COL neuronas de entrada, 1 neurona oculta y 2 de salida.

Variables Capa entrada:

Nota: para filas [desde 1 hasta i ésima] y columnas [desde 1 hasta j ésima]

n_0: $número\ de\ neuronas\ entrada = COL$

$$X_1[n_0, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_0 \end{pmatrix}$$

$$X1\_step\_X_{offset}[n_0, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_i \end{pmatrix}_{n_0 \cdot 1}$$

$$X1\_step\_X_{gain}[n_0, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_i \end{pmatrix}_{n_0 \cdot 1}$$

$$X1\_step\_X_{Ymin}[n_0, 1] = \begin{pmatrix} -1 \\ \vdots \\ n_i \end{pmatrix}_{n_0 \cdot 1}$$

Variables Capa Oculta:

n_1: $número\ de\ neuronas\ capa\ oculta$=1

$b_1[1,1] = (\vdots)_{1 \cdot 1}$

$IW1\_1[1\ , n_0] = (n_{1,1} \quad \cdots \quad n_{1,j})_{1 \cdot n_0}$

Variables capa salida:

n_2: $número\ de\ neuronas\ capa\ salida = 2$

$b_2[2,1] = \begin{pmatrix} \vdots \\ \vdots \end{pmatrix}_{2 \cdot 1}$

$$LW2[2\ ,1] = \begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1}$$

Programa principal:

$$X_1 = X_1 - X1\_step\_X_{offset}[n_0, 1] = \begin{pmatrix}\vdots\\ \vdots\\ n_0\end{pmatrix}_{n_0\cdot 1} - \begin{pmatrix}\vdots\\ \vdots\\ n_i\end{pmatrix}_{n_0\cdot 1} = \begin{pmatrix}\vdots\\ \vdots\\ n_i\end{pmatrix}_{n_0\cdot 1}$$

$$X_1 = X_1 \cdot (X1\_step\_X_{gain}[n_0, 1])^T = \begin{pmatrix}\vdots\\ \vdots\\ n_0\end{pmatrix}_{n_0\cdot 1} . (\cdots \quad \cdots \quad n_0)_{1.n_0} = \begin{pmatrix}\vdots\\ \vdots\\ n_0\end{pmatrix}_{n_0\cdot 1}$$

$$X_1 = X_1 + X1\_step\_X_{Ymin}[n_0, 1] = \begin{pmatrix}\vdots\\ \vdots\\ n_0\end{pmatrix}_{n_0\cdot 1} + \begin{pmatrix}-1\\ \vdots\\ n_i\end{pmatrix}_{n_0\cdot 1} = \begin{pmatrix}\vdots\\ \vdots\\ n_0\end{pmatrix}_{n_0\cdot 1}$$

Como se puede ver la variable x1 se reutiliza para ahorrar memoria.

$$Multiplicar\_array = IW1\_1[1, n_0]\ .\ X_1 = (n_{1,1} \quad \cdots \quad n_{1,j})_{1\cdot n_0} \cdot \begin{pmatrix}\vdots\\ \vdots\\ n_0\end{pmatrix}_{n_0\cdot 1} = (\vdots)_{1\cdot 1}$$

$$a_1 = Multiplicar\_array + b_1[1,1] = (\vdots)_{1\cdot 1} + (\vdots)_{1\cdot 1} = (\vdots)_{1\cdot 1}$$

$$a_1 = \frac{2}{1 + e^{-2\cdot a_1}} - 1 = \frac{2}{1 + e^{-2\cdot(\vdots)_{1\cdot 1}}} - 1 = (\vdots)_{1\cdot 1}$$

Se reutiliza la variable a1

$$LW2[2\ ,1] = LW2[2\ ,1] \cdot a_1 = \begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1} \cdot (\vdots)_{1\cdot 1} = \begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1}$$

$$LW2[2\ ,1] = LW2[2\ ,1] + b_2[2,1] = \begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1} + \begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1} = \begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1}$$

$$b_1[1,1] = \max de\ LW2 = (\vdots)_{1\cdot 1}$$

$$LW2[2\ ,1] = LW2[2\ ,1] - b_1[1,1] = \begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1} - (\vdots)_{1\cdot 1} = \begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1}$$

$$LW2[2\ ,1] = e^{LW2} = \exp\begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1} = \begin{pmatrix}\vdots\\ \vdots\end{pmatrix}_{2\cdot 1}$$

$$b_1[1,1] = \left(\sum_{i=1}^{2} (LW2)_i\right)_{1\cdot 1}$$

$$LW2[2,1] = LW2[2,1] \div b_1[1,1] = \begin{pmatrix} \vdots \end{pmatrix}_{2\cdot1} \div \left(\sum_{i=1}^{2} (LW2)_i \right)_{1\cdot1} = \begin{pmatrix} \vdots \end{pmatrix}_{2\cdot1}$$

Se reutiliza la variable LW2 y b1 para reducir al máximo la memoria.

La solución final es:

$$LW2 = \begin{pmatrix} \vdots \end{pmatrix}_{2\cdot1}$$

A.1.1 Estructuras utilizadas:

- Arrays de tipo float cuyo tamaño es de 24 bits y permite trabajar con números decimales. Se expresa de la siguiente manera:
  $$float\ x1[n_0] = \{\lambda_1, \dots, \lambda_{n_0}\};$$
- Funciones de tipo float donde las variables que utiliza son definidas en la cabecera del programa y se expresa de la siguiente forma:
  $$float\ tansig\_apply(float\ parametro)$$
  Para llamar a la función dentro del código principal se escribirá:
  $$a1[0] = tansig\_apply(una\ variable);$$
  Y automáticamente accede a la parte del código que contiene esa función y devuelve el resultado de ese código en esa función. Se expresa de la siguiente manera:
  $$float\ tansig\_apply(float\ parametro)$$
  $$\{Codigo;\}$$
- Uso de funciones tipo void (vacío), estas función no devuelve ningún valor concreto al código principal, simplemente se utilizan para cambiar el valor de alguna variables del código principal:
  En la cabecera se define la función:
  $$void\ divi\_arrays(float\ variable1, float\ variable2)$$
  En el código principal se llama:
  $$divi\_array(var1, var2);$$
  Para acceder al código hay que definirlo de la siguiente manera:
  $$void\ divi\_arrays(float\ variable1, float\ variable2)$$
  $$\{Código;\}$$
- Uso de funciones tipo vacío con punteros:
  Cuando en el código principal se llama a la función aparece delante de las variables el siguiente símbolo (&):
  $$resta(\&var1, \&var2)$$
  $(\&var1)$ Me da la dirección de memoria de datos donde se almacena la variable var1 con referencia al origen.
  Cuando se define la función en la cabecera, esta se indica de la siguiente manera:
  $$void\ resta(float\ *variable1, float\ *variable2)$$

Con el (*) estamos indicando que la variable1 es de tipo puntero y también que (*variable1) es el dato de la dirección dada por &var1. Y que recorriendo con la variable puntero (variable1 [0],…variable1 [1]) vamos sacando los datos de las distintas posiciones de var1 en función del tipo de variable.

A.2 ANN para 3 neuronas entrada, 1 neurona oculta y n2 de salida

Variables Capa entrada:

Nota: para filas [desde 1 hasta i ésima] y columnas [desde 1 hasta j ésima]

n_0: $número\ de\ neuronas\ entrada = COL$

$$X_1[n_0, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_0 \end{pmatrix}$$

$$X1\_step\_X_{offset}[n_0, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_i \end{pmatrix}_{n_0.1}$$

$$X1\_step\_X_{gain}[n_0, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_i \end{pmatrix}_{n_0.1}$$

$$X1\_step\_X_{Ymin}[n_0, 1] = \begin{pmatrix} -1 \\ \vdots \\ n_i \end{pmatrix}_{n_0.1}$$

Variables Capa Oculta:

n_1: $número\ de\ neuronas\ capa\ oculta$

$b_1[1,1] = (\cdots)_{1\cdot 1}$

$IW1\_1[1\,, n_0] = (n_{1,1} \quad \cdots \quad n_{1,j})_{1\cdot n_0}$

Variables capa salida:

n_2: $número\ de\ neuronas\ capa\ salida$

$$b_2[n_2, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_i \end{pmatrix}_{n_2.1}$$

$$LW2[n_2,1] = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1}$$

Programa principal

$$X_1 = X_1 - X1\_step\_X_{offset}[n_0,1] = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1} - \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_0.1} = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_0.1}$$

Como se puede ver se pisa la variable x1 para ahorrar memoria.

$$X_1 = X_1 \cdot (X1\_step\_X_{gain}[n_0,1])^T = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1} . (\cdots \quad \cdots \quad n_0)_{1.n_0} = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1}$$

Se vuelve a pisar la variable X_1:

$$X_1 = X_1 + X1\_step\_X_{Ymin}[n_0,1] = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1} + \begin{pmatrix} -1 \\ \vdots \\ n_i \end{pmatrix}_{n_0.1} = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1}$$

Se vuelve a pisar la variable X_1:

$$Multiplicar\_array = IW1\_1[1,n_0] \,.\, X_1 = (n_{1,1} \quad \cdots \quad n_{1,j})_{1 \cdot n_0} \cdot \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1} = (\dots)_{1 \cdot 1}$$

$$R_1 = Multiplicar\_array + b_1[1,1] = (\dots)_{1 \cdot 1} + (\dots)_{1 \cdot 1} = (\dots)_{1 \cdot 1}$$

$$R_1 = \frac{2}{1 + e^{-2 \cdot R_1}} - 1 = \frac{2}{1 + e^{-2 \cdot (\dots)_{1 \cdot 1}}} - 1 = (\dots)_{1 \cdot 1}$$

Se pisa la variable R1

$$R_2 = LW2[n_2,1] \cdot R_1 = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1} \cdot (\dots)_{1 \cdot 1} = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1}$$

$$R_2 = R_2 + b_2[n_2,1] = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1} + \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1} = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1}$$

Se pisa la variable R2

$$R_2 = R_2 - (\max de\ R_2) = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1} - (\max de\ R_2)_{1\cdot 1}$$

Se pisa la variable R2

$$R_2 = e^{R_2} = \exp \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1} = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1}$$

Se pisa la variable R2

$$R_2 = R_2 \div \left(\sum_{i=1}^{n_2} (R_2)_i \right) = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1} \div \left(\sum_{i=1}^{n_2} (R_2)_i \right)_{n_2 \cdot 1} = \begin{pmatrix} \vdots \\ \boldsymbol{n_i} \end{pmatrix}_{\boldsymbol{n_2.1}}$$

La solución final es:

$$R_2 = \begin{pmatrix} \vdots \\ \boldsymbol{n_i} \end{pmatrix}_{\boldsymbol{n_2.1}}$$

A.2.1 Estructuras utilizadas:

La variación respecto al código anterior es que se ha creado una variable salida en la que cambiando el valor de esta se varía la dimensión de los array:

$$\#define\ SALIDAS\ 2$$

Cuando escribimos define estamos indicando que salidas es una etiqueta y su valor es 2, con lo cual cuando en el código aparece una mención a salidas se está indicando un 2.

A.3 ANN GENERAL

Variables Capa entrada:

Nota: para filas [desde 1 hasta i ésima] y columnas [desde 1 hasta j ésima]

n_0: $número\ de\ neuronas\ entrada = COL$

$$X_1[n_0, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_0 \end{pmatrix}$$

$$X1\_step\_X_{offset}[n_0, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_i \end{pmatrix}_{n_0.1}$$

$$X1\_step\_X_{gain}[n_0, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_i \end{pmatrix}_{n_0.1}$$

$$X1\_step\_X_{Ymin}[n_0, 1] = \begin{pmatrix} -1 \\ \vdots \\ n_i \end{pmatrix}_{n_0.1}$$

Variables Capa Oculta:

n_1: $número\ de\ neuronas\ capa\ oculta$

$$b_1[n_1, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_i \end{pmatrix}_{n_1.1}$$

$$IW1\_1[n_1, n_0] = \begin{pmatrix} n_{1,1} & \dots & n_{1,j} \\ \dots & \dots & \dots \\ n_{i,1} & \dots & n_{i,j} \end{pmatrix}_{n_1.n_0}$$

Variables capa salida:

n_2: $número\ de\ neuronas\ capa\ salida$

$$b_2[n_2, 1] = \begin{pmatrix} \vdots \\ \vdots \\ n_i \end{pmatrix}_{n_2.1}$$

$$LW2[n_2, n_1] = \begin{pmatrix} n_{1,1} & \dots & n_{1,j} \\ \dots & \dots & \dots \\ n_{i,1} & \dots & n_{i,j} \end{pmatrix}_{n_2.n_1}$$

Programa principal

$$X_1 = X_1 - X1\_step\_X_{offset}[n_0, 1] = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1} - \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_0.1} = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_0.1}$$

$$X_1 = X_1 \cdot (X1\_step\_X_{gain}[n_0, 1])^T = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1} . (\dots \quad \dots \quad n_0)_{1.n_0} = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1}$$

$$X_1 = X_1 + X1\_step\_X_{Ymin}[n_0, 1] = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1} + \begin{pmatrix} -1 \\ \vdots \\ n_i \end{pmatrix}_{n_0.1} = \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1}$$

Como se puede ver se reutiliza la variable x1 para ahorrar memoria.

$$Multiplicar\_array = IW1\_1[n_1, n_0] . X_1 = \begin{pmatrix} n_{1,1} & \dots & n_{1,j} \\ \dots & \dots & \dots \\ n_{i,1} & \dots & n_{i,j} \end{pmatrix}_{n_1.n_0} . \begin{pmatrix} \vdots \\ n_0 \end{pmatrix}_{n_0.1} = \begin{pmatrix} \vdots \\ n_1 \end{pmatrix}_{n_1.1}$$

$$R_1 = Multiplicar\_array + b_1[n_1, 1] = \begin{pmatrix} \vdots \\ n_1 \end{pmatrix}_{n_1.1} + \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_1.1} = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_1.1}$$

$$R_1 = \frac{2}{1 + e^{-2 \cdot R_1}} - 1 = \frac{2}{1 + e^{-2 \cdot \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_1.1}}} - 1 = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_1.1}$$

Se reutiliza la variable R1

$$R_2 = LW2[n_2, n_1] \cdot R_1 = \begin{pmatrix} n_{1,1} & \dots & n_{1,j} \\ \dots & \dots & \dots \\ n_{i,1} & \dots & n_{i,j} \end{pmatrix}_{n_2.n_1} \cdot \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_1.1} = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1}$$

$$R_2 = R_2 + b_2[n_2, 1] = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1} + \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1} = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1}$$

Se reutiliza la variable R2

$$R_2 = R_2 - (\max de\ R_2) = \begin{pmatrix} \vdots \\ n_i \end{pmatrix}_{n_2.1} - (\max de\ R_2)_{1.1}$$

$$R_2 = e^{R_2} = \exp\begin{pmatrix}\vdots\\ \vdots\\ n_i\end{pmatrix}_{n_2.1} = \begin{pmatrix}\vdots\\ \vdots\\ n_i\end{pmatrix}_{n_2.1}$$

Se vuelve a reutilizar la variable R2

$$R_2 = R_2 \div \left(\sum_{i=1}^{n_2}(R_2)_i\right) = \begin{pmatrix}\vdots\\ \vdots\\ n_i\end{pmatrix}_{n_2.1} \div \left(\sum_{i=1}^{n_2}(R_2)_i\right)_{n_2 \cdot 1} = \begin{pmatrix}\vdots\\ \vdots\\ \boldsymbol{n_i}\end{pmatrix}_{\boldsymbol{n_2.1}}$$

La solución final es:

$$R_2 = \begin{pmatrix}\vdots\\ \vdots\\ \boldsymbol{n_i}\end{pmatrix}_{\boldsymbol{n_2.1}}$$

A.3.1 Estructuras utilizadas:

En este caso nos encontramos con matrices de dimensiones mayores que 1 tanto en filas como en columnas. Para poder llevar a cabo esta modificación se ha definido todo en un mismo array de la siguiente manera:

$$LW2[n_2, n_1] = \begin{pmatrix} n_{1,1} & \dots & n_{1,j} \\ \dots & \dots & \dots \\ n_{i,1} & \dots & n_{i,j}\end{pmatrix}_{n_2.n_1}$$

$$float\ const\ LW2\_1[n2 * n1] = \{X_{1,1} \quad \dots \quad X_{n2,n1}\}$$

Cuando se desea realizar la siguiente operación por ejemplo:

$$LW2[n_2, n_1] \cdot R_1 = \begin{pmatrix} n_{1,1} & \dots & n_{1,j} \\ \dots & \dots & \dots \\ n_{i,1} & \dots & n_{i,j}\end{pmatrix}_{n_2.n_1} \cdot \begin{pmatrix}\vdots\\ \vdots\\ n_i\end{pmatrix}_{n_1.1} = \begin{pmatrix}\vdots\\ \vdots\\ n_i\end{pmatrix}_{n_2.1}$$

Para multiplicar cada una de las filas de la matriz LW2 por el vector R1 se recorre mediante:

$$for\ (j = 0; j < n2; j++)$$

$$R2[j] = multiplicar\_arrays(\&R1, \&LW2\_1[j * n1], n1, 0);$$

También se han utilizado variables de tipo const, para que estas pasen a almacenarse de la memoria de datos a la memoria de programa y así descargar la de datos.